programmer's journal

The Resource Journal for IBM PC Programmers

CONSULTING

enhancements

communication

bugs·money·deadlines

Volume 2 Number 1

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

programmer s

READ THIS...

NOW, I KNOW I CAN MAKE BIG MONEY WRITING AND SELLING MY PROGRAMS. THIS BOOK TOLD ME WHAT TO WRITE — WHO TO SELL IT TO — THOUSANDS OF NAMES, ADDRESSES, IDEAS, GUIDELINES. "SOFTWARE WRITER'S MARKET" IS A FANTASTIC BOOK!
PROGRAMMING MANUAL

* WHO TO SELL YOUR PROGRAMS TO
* THOUSANDS OF COMPANY NAMES AND ADDRESSES, WITH DETAILED LISTINGS SHOWING:
  (1) WHAT PROGRAMS PUBLISHERS ARE LOOKING FOR
  (2) HOW THEY WANT YOU TO SUBMIT YOUR PROGRAM
  (3) HOW MUCH THEY PAY — AND WHEN!
* 100 CATEGORIES — FROM "ACCOUNTS RECEIVABLE" TO "GAMES" TO "VIDEO CONTROL" PROGRAMS
* HOW TO WRITE CLEAR DOCUMENTATION
* DEBUGGING TECHNIQUES

Order Shipped Immediately

Enclose check or money order for $19.95 (No C.O.D.'s) to:

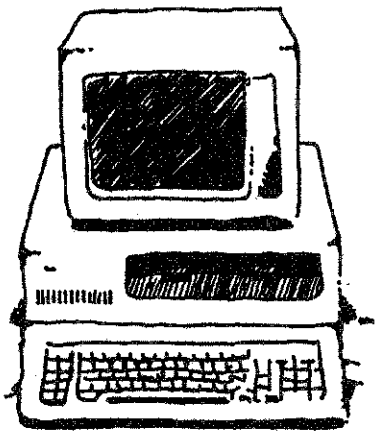
IPF Publications
146 Country Club Lane
Pomona, NY 10970
(914) 354-5585

Name .....

Address .....

City..... State..... Zip.....

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



PROGRAMMER'S JOURNAL

CONTENTS

HOUSEKEEPING 2

LETTERS 3

PJ PROFILE: Interview with Darrell Jones & Rich Rohde 5
How to handle freelance consulting work

PJ FORUM: DTACK GROUNDED/FNE 8
The inside story on the 8088 and 68000

C: SORTING IN C / Ken Smolek 13
Standard sorting techniques using C

PASCAL: PC WRITE & QUICKSOFT / Bob Wallace 20
A full-featured text editor with source code

VODS: VIDEO OPERATOR'S DISTRESS SYNDROME / Charles Wallach 27

BASIC: COMPILED BASIC - THE NEXT STEP / Ron Moak 29
Writing line-numberless structured code

ASM: ASSEMBLY MODULES FOR C / Ted Reuss 32
A custom support library for C

dBASE II: DATABASE RELATIONS / Jim Bohle 34
Designing flexible database relationships

UTILITIES: ULTRA UTILITIES / FreeSoft 41
Beyond the Norton Utilities

SOFTWARE REVIEWS
STYLE & PUNCTUATION / Richard Wanderman 42

PJ LIBRARY 44

MISC. 45

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HOUSEKEEPING

HOUSEKEEPING

The award for "Outstanding Programmer of 1983" is a hotly contested one. There are a number of quality software products on the market, and we each have our favorite contenders for this title. We could probably agree on several that would appear in the top ten: Prokey, Norton Utilities, PC-TALK, Personal Editor, and Lotus 1-2-3. If you ask me who has my vote for the inside track, I would say William Modlin and David Glickman. They wrote Word Proof, a spelling checker and on-line Thesaurus package from IBM.

Most of us have used spelling checkers. They save us from our worst enemy, ourselves. I KNOW how to spell hemmorage, or is it hemorage? (It's actually hemorrhage.) Word Proof did to spelling checkers what Lotus 1-2-3 did to spreadsheets. Best of all, Word Proof is only $60, another point for IBM.

Word Proof is a quality piece of software. Well thought out, well put together, easy to use (throw the manual away), and flexible. The minimum amount of prompts and menus to provide you with the maximum amount of features. The package is valuable as a "classic" piece of first rate programming. Briefly, Word Proof is a mini text-editor reminiscent of the Personal editor, with a spelling checker and synonym dictionary. You may edit and re-edit your text at will, start the spell check at any point in the text. Word Proof checks your spelling as you watch. It flips through the pages of your file one-by-one. A mis-spelled word is automatically replaced with the correct one you select. You may bounce into the text any time you wish for an alteration. A superior aspect of the checker is its word choice. If it finds a word like "tersly," it suggests "tersely, trebly, tearily, tensely, trestle, trembly." Most spelling checkers take "ters" and pull out some ending. You can be extremely far off on the spelling and it will find the right one. It also has a single word "immediate" check. Word Proof is especially nice for one-off letters that are a page in length that you do not wish to run through a mammoth dictionary. A "classic" piece of software. You can't go wrong with it.

Certain groups of programming support packages combine to form an especially nice software development setup. Memory-Shift with a RAM disk, BATCH files, compiler and editor working hand-in-hand can greatly increase your output. Describe the "ideal" environment that you are currently using so other programmers can compare notes. If you have found some source code support libraries that you particularly like, spread the good word.

— Greg

PROGRAMMER'S JOURNAL

Editor: Greg Estes
Managing Editor: Steve Robinson
Contributing Editor: Jim Bohle
Production: Christine Baldwin
Copy Editing: Marilyn Reaves

Contributing Writers:
Ken Smolek
Ted Reuss
Ron Moak
Bob Wallace
Richard Wanderman
Jim Bohle
FNE
Warren Hillman

Publisher:
Mary Carol Smith
Avant-Garde Creations, Inc.

Business Office:
Avant-Garde Creations, Inc.
1907 Garden Ave.
Eugene, OR 97403
(503-345-3043)

Editorial Office:
2765 Potter St.
Eugene, OR 97405
(503-687-0398)

Mailing Address:
P.O. Box 30160,
Eugene, OR 97403

Programmer's Journal. Volume 2, Number 1.

Copyright (C) 1984 by Avant-Garde Creations, Inc.

Published bi-monthly.

Postmaster send address changes to:
Programmer's Journal
P. O. Box 30160
Eugene, OR 97403
(503) 345-3043

Subscriptions: Bi-monthly rates are $24 U.S.
($30 in Canada, and $40 elsewhere.)

IBM and Personal Computer are trademarks of International Business Machines, Armonk, New York
dBASE II is a trademark of Ashton-Tate.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LETTERS

STRUCTURED BASIC

I was disappointed in the business about structured programming. I enclose a short re-do of the author's *raison d'etre,* where he warns the world about the non-testing features of a FOR-NEXT loop. I'm sure there will be other comments about it. He should turn it into a decrementing loop (step-1) and it will test for the zero condition without the need for the IF-THEN conditional jump. I hope you will print the short program for those who may not be familiar with this technique. It will correct the misunderstanding that Barkley left in the reader's mind.

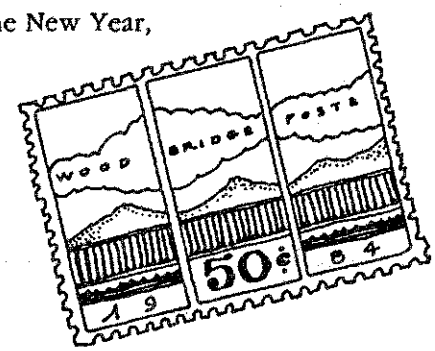
I believe there are many features of Structured BASIC that can be used in MS BASIC right now. First, the sub-routine arrangement - whether it's 10 or 50 lines - is immaterial (although it's hard to conceive a subroutine in BASIC running 2 pages long.) The important point, I believe, is that there should be no GOTOs that jump out of the subroutine, although they may be useful within the confines of the subroutine. Not all GOTOs are bad, but there are bad programmers who use GOTOs carelessly just as they use other commands incorrectly. People who get spastic about GOTOs sound like reformed alcoholics during the first few months of abstention.

It is sometimes necessary to juggle the main calling program to contain inter-subroutine GOTOs, if any are needed. The other complaint about subroutines not being able to accept parameters in BASIC is not really correct. All variables are available (global) to all subroutines. It's up to the programmer to make certain they are pure as the driven snow, correct and useful when a subroutine is called with a GOSUB. In fact, large subroutines can be tested with a prescribed range of variables, if anyone wants to set it up.

To show a Warnier diagram for such a simple subroutine is somewhat simple in itself. These are tools that are available, and it's up to the programmer to use whatever is necessary to produce safe intelligent code that works.

With best wishes for the New Year,

Lou Warner, P.E.
People Systems Ltd.
Woodbridge, CT




```
10 REM—- TESTLOOP —- SELF TESTING LOOP
20 CLS FOR K = 1 TO 15
30 READ QS
40 PRINT USING " ## ";K;QS;
50 PRINT TAB(10);
55 REM using a decrement loop eliminates the zero = 1
60 FOR J = QS TO 1 STEP -1
70 PRINT "*" ;
80 NEXT J
90 PRINT
100 NEXT
110 DATA 3,7,0,1,7,2,4,6,1,2,1,0,1,6,9
```

PROGRAMMER'S
CALCULATOR UPDATE

Regretfully in this world of software, bugs often creep in where you don't want them. So it was with the first few releases of the Programmer's Calculator. Fortunately it does not take long before the letters start coming in, pin-pointing the problems. As a result, Programmer's Calculator went through a few changes. If you are not sure you have the latest release, check for version 1.22 on the opening screen. The changes in the program over the original include:

version 1.2 Assembly language driver to allow instantaneous switching between screens

version 1.22 Faster loading of the screens (20 times faster then v 1.2)

version 1.22 Correction of problem when both color/graphics and monochrome monitor cards are installed

Unfortunately, a couple of the bugs in the original two releases made the program worthless for some people, because of their system configuration. The new release has been fully tested on computers with a hard disk, multiple displays and DOS 2.0. If you would like a new version of the program, send a formatted disk and mailer with return postage to:

Ronald Moak
Forest Computer Consultants
P.O. Box 1463
Springfield, OR 97477
Phone (503) 746-7470

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

WARNIER-ORR REVISITED

Paul Barkley stated in his November/December '83 column, PJ Forum, that he readily admitted that Warner Structured Diagrams don't work all that well in practice. Such a flagrant display of total ignorance concerning Warnier-Orr by Mr. Barkley is astounding given that your other authors stick to writing about things they are technically competent to discuss.

Please refer Mr. Barkley to the following firm so that he may become trained to a technically competent level in Warnier-Orr techniques:

Ken Orr & Associates, Inc.
1725 Gage Boulevard
Topeka, KS 66604
(800-255-2459)

For the benefit of your readers who were unfortunately misled by Mr. Barkley's malpractice-based comments concerning Warnier-Orr, there are many advanced data processing installations enjoying spectacular success stories using Warnier-Orr for designing both small and large systems.

My staffs have been able to consistently implement systems with fewer than 0.1 logic errors per program following initial clean compiles, without missing user specifications of any type. This experience is repeated in firm after firm that has correctly implemented Warnier-Orr. Further, the largest information systems consulting firm, Arthur Andersen & Co., has based its very successful Method/1 design methodology primarily on Warnier-Orr.

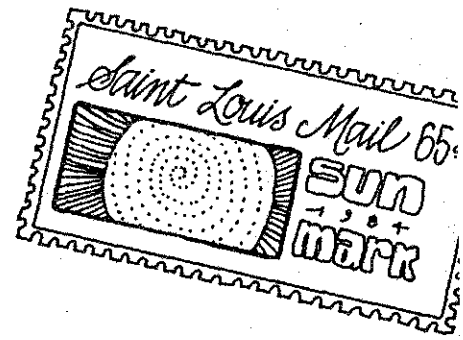
An article in another IBM PC magazine (about three months ago) noted similar results in using Warnier-Orr for designing small systems using BASIC. The author of that article stated that he was able to develop program after program, using Warnier-Orr, with no logic errors whatsoever in his code. He in fact could not recall when he had last encountered a logic error in his programs during the past two years, other than the one time that he did not use Warnier-Orr.

Mr. Barkley's primary error in the examples he provided was in not understanding that Warnier is based on data structuring, not logic structuring. His error is one commonly made by novices during the first few hours of attempting to use Warnier-Orr improperly. His personal results undoubtedly suffer due to his not correctly understanding the major difference between ineffective logic-based structured approaches and data structured approaches.

Warnier-Orr charts are a excellent - but quite different structured technique.

Cordially yours,

Sunmark, Inc.
Vince E. Heiker
MIS Director
St. Louis, MO



TURBO PASCAL

Enclosed is my check for $24 to renew my subscription through Dec. 1984. Please keep up the good work. I currently use two Pascals on the IBM PC: Microsoft 3.11 for number-crunching, because of its fast 8087 support & compatible Fortran compiler, and Turbo Pascal from Borland International, because of its fast compilation and excellent interactive debugging - just about as painless as using an interpreted language. Turbo Pascal is a bargain at $50 including the WordStar compatible programming editor - but there's a catch: although not mentioned in their advertisement, they want an additional $100 if the object code is to be used on machines other than the one for which the license was purchased.

Jeffery M. Speiser
San Diego, CA

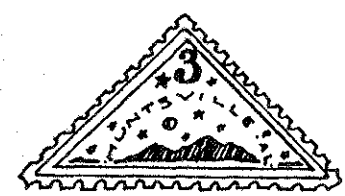
Turbo Pascal
Borland International
4807 Scotts Valley Dr.
Scotts Valley, CA 95066
ph. 227-2400 Ex 968
(CA call 800-772-2666 Ex 968)

TIDBITS

Please renew my subscription for another year. Appreciate the good tidbits in the Misc. section of the last issue. PJ is getting better every issue, keep it up.

P.S. I will contribute when I get caught up (if!!).

Robert Gillis
Huntsville, AL



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ PROFILE

CONSULTING

a discussion with Darrell Jones and Rich Rohde, computer system consultants.

PJ: How long have you been working with computers?

RICH: Since May of 1971.

DARRELL: I started programming in 1971, that makes about 12 years in data processing. I got out of graduate school in 1971 and it has been downhill ever since. Things have really changed during the years I have been doing consulting.

PJ: What do you usually charge for consulting services?

RICH: Currently I am charging $45 an hour for software development. If it is a problem I can solve in a couple of minutes (like over the phone) then there is no fee. As president of the local IBM PC User Group I have an obligation to help them when I can. If I have to go on-site and do something then it is $45 an hour.

DARRELL: Around $35 to $40 an hour. I do both consulting and systems work in a large mainframe shop. I work by the hour and do some contracts. If it is a contract, then I make sure all the requirements are known before I make a bid. This might mean that the client must spend some hourly money to get the analysis done before the bid is made. Data flow diagrams should be complete and the user in agreement as to what reports are going be produced, what processes are to done on the screen etc. No one is going to be happy if you make up a bid without an understanding of what you are selling.

PJ: If a person called you and said "I need an inventory system that will run on my IBM," how would you respond?

RICH: Well, we would have to go through the phases of software development and system analysis. But first of all I would ask if that person had done some investigation, have they searched for a package that already exists which would fulfill their needs.

If they do desire some custom software, then we would decide whether they would own the final package or whether I would retain ownership rights and could resell it to other companies. If they allowed me to retain the rights and the package seemed general enough, then I would probably just charge them what the market value was, what they would pay for the same package at a store. I would then market the software on my own or license it to a publishing company to recoup my investment. Normally I would go to a marketing agency before accepting the job to be sure the software had potential.

If the client said they want to retain ownership rights, then it would be a straight $45 per hour for development.

DARRELL: If it is just modifying a package or part of a pre-existing system, then I tend to do that hourly. Working hourly is less risky from the consultant's point of view. You could start a contract and find the user will run out of disk space immediately, or you can have a hardware crash the second day into the project. All of these things have happened to me. People in business tend to figure out the cost of a project based upon the shortest amount of time it would take if everything went right. If you are doing a flat bid on custom software, then you really need to pad out the figure based upon what happens in real life. You have to cover yourself, and then take a little extra for running the risk of a flat bid.

If you bid too low, then you will not stay in business and that doesn't do anyone a favor, especially the user. If you are not making a profit and therefore are not motivated to continue quality work, then you will lose interest or go out of business. You have to run consulting like any other business. The worst thing a user can do is to buy custom software from someone who is going to go belly-up within the year. It is almost endemic to the profession to underbid; you think, "gee I am going to find 5 other businesses that will want the same software, so I can package it and make my money back that way." Lose on the single job but regain it through volume.

PJ: Do you make the initial visit to a client at no charge?

RICH: Yes, that is possible. A courtesy call is usually feasible depending on the size of the project. The second or third visit is definitely a chargeable item.

PJ: Do you prefer short or long projects?

RICH: Short ones. Long projects do not give me the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

variety that I enjoy. The longer you work on one project the more boring it becomes. Something that runs a month or two months is about right.

DARRELL: The analysis is a large part of any project, and this is usually done by the hour. A contract involving a bid must be long term or it is not worth working up the specs. The length of the analysis depends on a number of things. If you do the systems work and find that a pre-packaged inventory system fills the bill then the analysis work is almost the whole charge. The analysis is more of an inflexible chunk of time. It needs to be done correctly whether the user ends up with a custom or pre-packaged system.

The tools are getting better for prototyping a software system. Screen generators and report generators allow you to create a mock-up version for the user to look at. On very large projects you definitely need to do this, an incremental approach to the implementation. If you don't, the analysis part just drags on too long, with no user involvement other than they tell you whether the data flow diagrams are right or wrong. I have found that users can understand data flow diagrams; indeed, they are the greatest thing since sliced bread. A flow chart is procedural, whereas a data flow diagram is non-procedural. A data flow diagram shows how processes relate to one another. Something like "the inputs to the payroll file will be the hours worked typed in at the screen plus the rates coming off this table and the outputs will be the paychecks" and so on. Flowcharts, it has pretty much been proven, have no value. There have been a number of studies that show whether a program turns out better if it has been flowcharted first or not. Data flow diagrams are a whole different kettle of fish.

PJ: If both sides, the users and the analyst, can understand the diagrams, they offer a tremendous asset.

DARRELL: They are pictorial. Too many program or system specifications are narrative. Words have value but if you show someone a picture then they can say "that definitely doesn't come from here."

PJ: You mentioned "incremental implementation."

DARRELL: The idea is to get something in place as soon as possible. You are always trying to complete a quality package in the shortest possible time. The user appreciates a payback as the project goes along rather than just at the end. Incremental implementation, bringing up a working part of the final solution as soon as is feasible, is helpful with the analysis as well. The user then has a real piece of the system to bang away on. Data processing projects historically tended to wait until every piece was totally complete before it was delivered. Large systems could take years to implement. By th time you were done the technology had change drastically. The 80/20 rule is valid here. You can get 80% of the work done in 20% of the time.

PJ: Do you rate the client as to whether you even wish t do business with them? Are novice or experienced user better?

RICH: I haven't rated clients too much in the past, but i is becoming more of a reality. I try to check thei background and see if they are a financially stable type o person or company. This of course applies to the longe kind of projects.

The person's personality is important. Are they easy t work with? If the client is hard to deal with, then it i going to take a lot longer to get the final produc completed. Find out if they have the project pretty wel outlined to start with. Are they a STRUCTURED type o person. First time users are definitely less experienced but the ability to organize their thoughts is much mor important. Find out if their business is organized.

DARRELL: I talk with a client and find out what the know about their business from an analytical point o view. They may know their business quite well but no have thought it out logically. Any previous experienc with a computer is helpful. They need to know wha they WANT. If they need only a piece of a system ther this is not so crucial, but if they are asking for a comprehensive package, understanding their busines flow is essential. An experienced user is more realistic about what to expect. They also can begin relating thei everyday tasks to a computer, they know they are goin to save substantial time when their work i computerized.

PJ: How do you handle bugs and requests fo enhancements?

RICH: I usually give them software status reports to fil out if a bug shows up. And I also give them an erro message listing so they know what to expect.

Enhancements or "incorrect" system results are more o a problem if you have not done the analysis correctly, meaning determine the clients needs, the data flow, etc. If you skipped this part and the client says "Oh no, that' not right" then you shouldn't charge them for it. But i you do everything properly, and the client comes an requests a major change then you have to make a determination. Evaluate the change; if it's a mistake i the systems analysis, then it is your responsibility otherwise, it is chargeable to the client. If it was a unforeseen change, then the cost may be spread amon both of us.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DARRELL: If you are doing a bid, then everything must be preferably down on paper. Incremental enhancements are dealt with as they come up. There is incentive on both sides to be as clear as possible from the start. The client may not realize it, but they want you to complete the project as soon as possible too. If they are willing to take the delay and the expense, then the enhancement can be worked in. Is it worth it for them? There is a tendency for users to hit you with a bunch of enhancements when you are 80% done with a system. Psychologically this is tough. Preferably they will allow you to get the system up and running so they can prioritize any additional improvements.

A project is at its most vulnerable when it is 80% done. Most of the work is completed with the least to show for it. Incremental implementation can help this. It is a hump that both sides have to get over.

PJ: How do you handle the client after the package is finally complete? Do you feel a need to cut off the weekly phone calls?

RICH: Well, you do not totally sever the connection. It is important to maintain a line of communication. This is beneficial for both of you. Your customers feel that you are there supporting them, and you have a good reference. The only exceptions are the rare clients that are more trouble then they are worth.

DARRELL: Support is just a part of the job. This presents no problem if you are charging by the hour. On a bid then you look at satisfying the contract. If the phone calls get too out of hand, then you may have chosen the wrong client. I have never had that happen.

PJ: Rich, you are marketing a system for the IBM PC, aren't you?

RICH: Yes, I have developed an orthodontic system that runs on the IBM PC. It is being marketed on the Pacific coast. In fact, tomorrow I will leaving for a show in Texas that will be featuring my system at a symposium for orthodontists. I do not have anything to do with the marketing of the package. I am responsible for most of the documentation, enhancements, and any updates. Officially I own the copyright to the software and the marketing company is licensed to sell it. My royalties from the sale of systems go into an account which I bill at $45 an hour. It will take some time to regain my initial investment in the software.

The Orthodontic System is basically an accounts receivable package that handles regular billing and insurance billing. Two or three orthodontists that share an office can use the same system together.

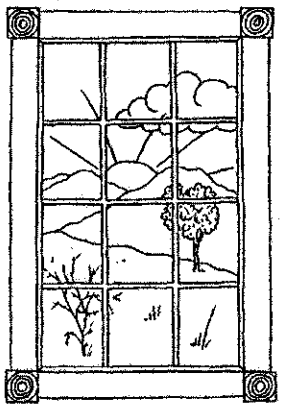
PJ: Any particular recommendations for other programmers who plan to enter the consulting field?

RICH: Someone thinking about doing consulting work should definitely get some business background. Go to school and take some business courses. Learn how to consult. □

For more information about the Orthodontic System contact:

Rich Robde
12260 SW 124th
Tigard, OR 97223
(503-620-6862)

NOW FOR THE IBM PC + COMPATIBLES



WINDOWS FOR C™

SCREEN MANAGEMENT TOOL FOR C PROGRAMMERS

A COMPLETE WINDOW DISPLAY SYSTEM

- Unlimited windows and text files
- Instant screen changes
- Horizontal and vertical scrolling
- Variable text margins
- Word wrap option
- ASCII file handling routines (including tab expansion)

BAR GRAPH ROUTINES INCLUDED

FOR ALL DISPLAY TASKS

FAST
Line-oriented ASM subroutines

COMPREHENSIVE
Access to all text mode capabilities

VERSATILE
Building block subroutines

FULLY DOCUMENTED

Available For: Lattice C, C 86, Microsoft C, DeSmet C

Introductory Special $119.95
($150 after May 31)
Demo disk & manual $30
(Applies toward purchase)

A PROFESSIONAL SOFTWARE TOOL FROM
CREATIVE SOLUTIONS
21 Elm Ave., Box P2, Richford, VT 05476

For Order or Information: 802-848-3804
Master Card & Visa Accepted
Shipping $2.50
VT residents add 4% tax

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ FORUM

A FABLE FOR OUR TIME

by FNE, writer of DTACK GROUNDED, The Journal of Simple 68000 Systems, covering among other things the 68000 board for Apple computers. The following excerpts are from issues #16 and #25.

Once upon a time two young men founded a software company to write a BASIC interpreter for a new invention called a personal computer. Because everyone stole their software, their software became the standard for the new industry. As the new industry grew, so did the software company so that the software company continued to be the standard for the industry.

For many years this company prospered mightily by providing software, especially BASIC interpreters, which ran in 8-bit microprocessors. One day a blue giant appeared out of the forest and requested software for the 8-bit 8088. Since the 8088 is source compatible with the 8080, the software company was easily able to respond favorably to the giant's request. And the treasury of the software firm grew more rapidly as royalties from the giant poured in.

Since the giant's machine used an 8-bit processor and ran software written for yet another 8-bit processor, the giant's machine ran at about the same speed as earlier 8-bit machines supported by the software company's products. This greatly puzzled many people since an evil sorcerer had cast a spell on the land so that the people would believe that the 8-bit machine was really a 16-bit machine.

The people resolved this problem by asserting that the giant's machine was a 16-bit machine (although it was not) and that it therefore ran swiftly (although it did not). But the two young men had amassed enough wealth to purchase an antidote for the evil sorcerer's spell and were aware of the true facts.

Among the true facts they were aware of was the fact that 99.999% of their (substantial) income came from 8-bit machines running at 8-bit machine speeds.

This posed a dilemma one day when 99 pygmies and one large oaf turned up and asked the software house to provide a BASIC interpreter for THEIR machines, all of which were based on a real, honest 16-bit processor called the 68000. The dilemma was caused by the fact that the 68000, being a true new generation 16-bit processor, was 10 to 20 times the speed of the machines from which their (substantial) income derived.

And if those 99 pygmies (and one large oaf) began manufacturing machines which could run 10 or 20 times faster than those 8-bit machines, then the sales of the 8-bit machines would doubtless diminish, and their (substantial) income would ALSO diminish. And the blue giant who had been friendly might become less so.

But the two young men and their assistants came up with a plan: they would produce a BASIC interpreter for those 100 68000-based machines which would run at EXACTLY THE SAME SPEED as all the 8-bit machines. And then they realized that the problem would recur as even more real, honest high performance 16-bit processors became available. The two young men were no dummies and they came up with a generic solution:

They would write a version of their BASIC interpreter in a sort of 'p-code'. Not literally PASCAL p-code but that sort of thing. They would then write native code for each of the real, honest 16-bit processors to interpret the 'p-code'. Therefore, TWO levels of interpretations would be involved and the real, honest new-generation 16-bit machines would run as slowly as all those older 8-bit machines.

And the (substantial) income of the software house remains safe and the blue giant remains friendly.

End of fable.

CRY FOUL!

One reader (of DTACK GROUNDED) made a point of calling us up shortly after receiving issue #24 and strongly protested our (mis)treatment of UNIX. His biggest complaint was that we were criticizing UNIX without having lots of personal hands-on experience.

How much experience do we have to have to note that UNIX has an exceptionally complex command structure, and that it requires a great deal of time to learn? Or to note that a really good UNIX system costs $35,000 with one work station? Or that the UNIX REVIEW contains no information directed to the UNIX user - the person who gets his/her hands dirty on the keyboard?

A philosophical question: must one be able to play the violin, flute and flugelhorn to be a music critic? Does one have to be a best-selling novelist to be a literary critic?

It is ironic that our conversation occurred on the day Electronic News arrived with a LONG article on upper-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

end small computer operating systems. The article was focused around Jean Yates, who is a UNIX person. Jean says UNIX is not doing as well as she had earlier expected. The article also asserted that the mainframe folks are getting really interested in UNIX. You betchum! UNIX, like a mainframe, is something that gets done to you whether you like it or not. The poor sucker hit with that 600-page manual has the same freedom of choice as a keypunch operator.

Actually, our comments on UNIX are based not only on our own personal observations, but on consultations with a couple of heavy-duty, gung-ho UNIX types, the ones who use their UNIX-based computer 8 hours a day, five or six days a week. Neither of them will reveal whether their employer permits them to break for lunch. These types seem to believe (as does the irate caller, we believe) that UNIX will arrive, eventually, but that you will not recognize it.

They described to us yet another 'shell' which makes a UNIX system as simple for a clerk to use as, say, the operating systems of the Pets/Apples/Trash-80s. Our question: if we get such a simplified operating system, how come we still have to give Western Electric a 70-pound bag of gold for permission to use it?

If you are a UNIX fan, here is something that might worry you: Jean had her market forecasting firm, Yates Ventures, project sales of the various 16-bit operating systems up through 1986. Jean, who is very definitely a UNIX person, was able to predict that UNIX would do better than only ONE operating system: Pick! She thought all the others would do MUCH better - and so do we! Who says we never agree with the UNIX experts?

One supposes that Jean has not noticed that Pick is the ONLY operating system in the group that is more expensive than UNIX. One also supposes that Jean has not noticed that the two 16-bit operating systems which she is predicting will have the fewest installations are the two which are purchased by honchos for use by peons, while all the others, which she is predicting will be most successful, are purchased by the person(s) who will use them.

Fifteen years ago 100% of all computers purchased were purchased by honchos for peons to use. Current predictions are that 1985 will be the year that personal computer purchases will surpass mainframe purchases in absolute dollar volume. If UNIX is going to become successful it had better become successful FAST because the clear trend is away from computers bought by honchos for peons, which is what UNIX is about. Make that a clear, RAPID trend!

If you are selecting an operating system for your own use, do you want one which is optimized for a single user, you, or one which is optimized to share a single CPU amongst a number of workers in an office? Is this a real choice?

BYTE WATCH:

First, we want you to know that we did NOT write the letter in Oct. '83 BYTE magazine about the Tandy Model 16 68000 BASIC being ungodly slow. We loved the official reply from Tandy, which never once mentioned that the BASIC was written in C.

Oct. BYTE has about nine articles on UNIX. One is by David Fiedler, who writes a newsletter for the UNIX folks. On page 152, David argues that any operating system which can only support a single user is worthless and is to be rejected in the search for a universal operating system. He seems to think that MS-DOS 2.0 is a win for UNIX. It is absolutely amazing how wrong-headed newsletter writers can sometimes be.

First, 1985 is the year when personal computers (that's the single-user kind, David) are going to pass mainframes and minis combined in total dollar volume. With personal computers moving up (the 68020 will be available over the counter in 1985, and that chip is a genuine VAX-killer!) and mainframes moving down (have you seen the latest IBM releases?) computers in the $30,000 - $300,000 area, which is UNIX's niche, are gonna get squeezed badly.

Second, David seems to feel that MS-DOS, which has adapted hierarchical file structure (a noted UNIX feature), will naturally support multiple users, beginning tomorrow. Therefore, MS-DOS is really UNIX! Sigh. Look, guys, hierarchical files are no more an exclusive property of UNIX than threaded code is an exclusive property of FORTH. A hierarchical file structure is REQUIRED once enough files are at hand.

Don't you readers keep your floppy disks with text files separate from the BASIC programs which are kept separate from...? That's a hierarchical file structure, guys! If you have a big hard disk you will naturally place all your files on that big hard disk and then your hierarchical structure has to be in the disk operating system. Did we ever tell you that real UNIX REQUIRES a big (and fast) hard disk? To repeat, the hierarchical file structure is required by the big disk and is not an exclusive property of UNIX.

Third, MS-DOS is written in assembly, NOT C! As a result, MS-DOS gives excellent performance with a relatively inferior microprocessor whereas UNIX, written in C, requires the best microprocessor available

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

in order to provide adequate performance.

Do not kid yourself that Bill Gates does not know the difference in performance between an operating system (or BASIC) in assembly and one written in C. It is absolutely no accident that the operating system (and BASIC) which Microsoft wrote to support its highly profitable relationship with IBM are written in assembly, and that its 68000 BASIC is written in C.

THE OP CODE FIELD:

In the 6502, all op codes are one byte long. The vanilla 6502 has just 156 instructions, which is a good fit for a one-byte op code field. We believe - although we are not certain - that all of the 8080 op codes were one-byte codes. (Naturally, both microprocessors had many instructions which carried along parameter fields so that the total instruction might be two or three bytes. Nevertheless, the op code is just a single byte.)

When Intel decided to make an early move into the 16-bit arena, they were limited (as all chip makers were and are) in the amount of resources, such as registers, that they could put on the chip. And, they wished to maintain COMPLETE functional compatibility with the 8080 so that code written for the 8080 could be translated into 808X code. As a result, the 808X has many op codes which are one-byte codes.

We trust it is apparent that if one has 256 one-byte op codes then one cannot have ANY op codes which are longer than one byte? And that 255 potential 16-bit op codes are permanently lost for each one-byte op code? And that devices which have some one-byte op codes can have much more compact code than a device which has 16-bit op codes as the shortest available?

If you have 16 registers, not 231, you can get by with a 4-bit register address field. Since the 808X has only 25% of the internal resources of the 68000, it can get by with one-byte op codes. That's more efficient. Since the Wang 500 has only 16 registers it can get by with only a 4-bit address field. That's more efficient.

THOSE 64K CODE MODULES:

In the past 15 years, almost all computer scientists in universities have been limited to hands-on experience with minicomputers having a 16-bit address field. When memories larger than 64K came along, segmentation was the only way to use those larger memories. Code was indeed written in 64K (or smaller) modules, but out o absolute necessity rather than as a law of the natura universe. If one is working with the PDP11/70 or an Inte 808X, one is STILL limited to the necessity of coding ir 64K modules. The argument presented by the Intel type was that this was more efficient (correct) and therefore desirable (incorrect).

As we reach and proceed beyond each boundary o address space it is inevitable that there will be a backlash, a natural resistance to continuing onward. Four bits are enough to address 16 registers but it is highly inefficient to use eight bits to address 17 registers! The inefficiency of moving to larger address fields (or op codes) is REAL! It is also NECESSARY if we are not going to program with 16 registers and 320 program steps forever. Many o the arguments presented by the Intel type were absolutely correct but the point of the argument, that the 808X was superior to the 68000, was dead wrong.

The 8088 has an 8-bit data bus, 8-bit op codes, 16 bytes of internal registers and a longest operand of 16-bits. The 68000 has a 16-bit data bus, 16-bit op codes, 64 bytes of internal registers and a longest operand of 32 bits. It is true that both the 808X and the 68000 have 16-bit multiplies and divides. Based on these facts, that Intel type had the nerve to assert that the 8088 and 68000 were both 16-bit microprocessors, the SAME size! Unbelievable!

ARGUMENTS FOR THE 808X

As we have said, efficiency is a REAL issue. Moving just slightly beyond 64K is problematic, as many of you with Cram-apples or Legend Industries memory expansion in your Apples have discovered. In our opinion, the Intel 808X architecture is indeed optimum for memory configurations up to 192K. That gives you 64K for your machine code, 64K for your high level language code, and 64K for your high level language data. In our opinion the 808X is the right 'engine' for the 'over 64K' to 'under half a meg' machines.

RUMORS MONGERED HERE:

There are a few chip designers and sellers at Intel (the rumor goes) who would like to shoot Bill Gates right now. It seems the Microsoft folks can't read, and as a result Intel has a very large pile of 80188s it can't ship. And Intel is redesigning the 80188 chip. Again.

It's like this: the 8088 spec sheet reserves two of the 256 jump vector addresses for future Intel use. Microsoft

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

went ahead and used them in the MS-DOS operating system anyway. The large pile of 80188s that Intel can't ship use those two reserved vectors for a hardware purpose. Understand, now, Intel was right and Microsoft was wrong. And: true, Microsoft could modify MS-DOS to comply, retroactively, with Intel's entirely correct documentation.

Unfortunately, there are about 12,000 application programs sitting on computer retailer's shelves all over the country which call those vectors... It would seem that in this case Intel's substantial software support backfired.

Since Intel's documentation scrupulously documented that those two vectors are reserved, they are (the rumor goes) refusing to take back the 80188s they have sold, unless (the rumor continues) the customer uses a blue logo with three alphabetic characters.

And now you know why Peanut has not been shipped yet, and what CPU Peanut uses. We wonder how long it will take Intel to change the mask - again - and get the chip back into production - again? (Do you realize that most folks don't know why Peanut hasn't been released but YOU do?)

And you also know why the Intel folks would like to shoot Bill Gates. (If the rumor is correct.)

HERE ARE THE FACTS:

The following information was gathered from the Intel iAPX 86/88, 186/188 User's Manual, Intel #210912-001, pages 4-10 and 5-5; and from the IBM PC XT Technical Reference Manual, page 2-4. Page 4-10, shows that the 8088 has 5 (five) dedicated interrupt pointers (#0 - #4), 27 reserved interrupt pointers (#5 - #31), and 224 (user) available interrupt pointers (#32 - #255).

Page 2-4 of the IBM manual reveals that MS-DOS uses 25 of those 27 reserved interrupt vectors. Specifically MS-DOS uses #5 and #8 thru #31 inclusive.

Now turn to page 5-6 of the Intel manual. Table 5-2

DESMET C $109

PCDOS+ — CP/M-86++ — MP/M-86++ — CCP/M-86++

- **C DEVELOPMENT PACKAGE**
  C Compiler, Assembler, Linker, and Librarian
  Full Screen Editor — Native 2.0 Support
- **COMPLETE IMPLEMENTATION**
  Everything in **K & R** (incl. **STDIO**)
  Small case — 64K code, 64K data/stack
  All 8088/8087 data types
  Intel assembler mnemonics
- **OUTSTANDING PRICE/PERFORMANCE**
  Aug. '83 *BYTE* "SIEVE" Benchmark
  135 bytes compiled — 6144 bytes linked
  65 sec. compile (fdisk) — 11.5 sec. run (10 iter)
- **SUPPORT LIBRARIES**
  Both 8087 **and** Software Floating Point
  PC Keyboard/Display
  Trig/Log functions
- **UPDATES**
  Updates and Bugs reported in newsletter
  No time/number limits

+ Trademark IBM ++ Trademark Digital Research

ORDER FORM

Name _____

Address _____

_____

_____

Phone _____

OS: ☐ PCDOS ☐ MSDOS ☐ CP/M-86, MP/M-86, CCP/M-86
DISKS: ☐ 8"(3740) ☐ 5.25" SS(160KB) ☐ 5.25" DS(320KB)

MACHINE? _____ (IBM, ZENITH, NEC, etc.)

DeSmet C Package ..... $109 _____

Manual Only (refunded with order) ..... $20 _____

Update Only ..... $20 _____

Sub Total _____

California Residents add 6 % Sales Tax _____

Send To: **C WARE**
P.O. Box 710097
San Jose, CA 95171-0097
(408) 736-6905

Total _____

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

reveals that the 80186/8 uses #5 as a vector for the array bounds exception, and also vectors #6 thru 15 and #18 and #19. Which means that MS-DOS programs will NOT, in general, run on the 80186/8. We are told that the better the program, the more likely that those MS-DOS vectors are being used at the application level.

The fact is that the 80186/8 is a very cost-effective chip at the hardware system level, as we have reported before. It is highly logical that IBM would, given enough lead time, use the 80188 in a highly cost-sensitive device such as a home computer. Let us assume that this tiny, trifling conflict was discovered very late in the design cycle for Peanut (in large companies, the hardware and software types sometimes don't communicate well). We can think of four ways out of the dilemma:

1) Cancel Peanut.

2) Go to market with a product that won't run all those MS-DOS-based PC programs out there.

3) Re-design Peanut to use the 8088 - an expensive and time-consuming alternative involving a complete re-layout of the main circuit board.

4) Get the company which you own a large piece of to take back all those 188 chips IBM (may) have bought and re-design the 188 to conform to MS-DOS. This is also time-consuming but the expense is placed elsewhere!

We would like to thank Otherwise Intelligent for the rumor and Harry S, who is affiliated with a networking firm, for his help in piecing the facts together.

CREDITS:

If you give somebody a photocopy of this please remind them that there will be more hardware information and ESPECIALLY more software information about the 68000/16081 combination in subsequent issues of DTACK GROUNDED.

THE FOLLOWING TRADEMARKS ARE ACKNOWLEDGED: Apple, II, II+, IIe, soft, LISA: Apple Computer Co. Anybody else need a 166th million ack, have your legal beagles send us the usual threatening note. (To the best of our knowledge, Apple has not trademarked Mackintosh yet. We don't have a THING planned for tomorrow; we just might mail something to the trademark registry office just for the hell of it. With trademarks, it's first come, first served.)

SUBSCRIPTIONS: Beginning with issue #19, subscriptions are $15 for 10 issues in the U.S. and Canada (U.S. funds), or $25 for 10 issues elsewhere. Make the check payable to DTACK GROUNDED. The address is:

DTACK GROUNDED
1415 E. MCFADDEN, STE. F
SANTA ANA CA 92705
(714-845-4884)

REDLANDS IS BACK? Yes! It's just that REDLANDS is about a hardware subject for the first, but not last, time. Who said there is nothing new under the sun? Next month we will provide some software demonstrating how to use the 16081 to calculate one or more transcendental functions and what the timing is compared to the Intel 8087. We also expect to provide, in the near future, comparisons of transcendental function (in)accuracy between the 8087, 16081 and our own 62-bit F.P. package (REDLANDS in issue #15 & #16).

To the best of our knowledge, nobody has ever looked at the (in)accuracy of the 8087's built-in transcendental functions except Dr. Dobbs' Journal, reported elsewhere in this issue. The results reported by Dr. Dobbs are, um, interesting. Stick around; it looks like we - that's you and us - are gonna have lots of fun!

Copyright (c) 1983 by Digital Acoustics, Inc.

□

TELECOMMUNICATIONS TYPESETTING

COMPUTER OWNERS: If you have a modem, text editor, and communications program, you should consider the improved readability and cost savings provided by *TYPESETTING* your program documentation, manuscript, newsletter, or other text instead of just reproducing it from line printer or daisy-wheel output. Computer typesetting offers you high quality, space-saving copy that creates the professional image you want! Hundreds of type styles to choose from with 8 styles and 12 sizes "on line." And it's easy to encode your copy with our designated typesetting commands.

COMPLETE CONFIDENTIALITY GUARANTEED
— bonded for your protection —
PUBLICATION DESIGN, EDITING, & PRODUCTION
Editing & Design Services Inc.
30 East 13th Avenue Eugene, Oregon 97401
Call Collect 503/683-2657

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SORTING IN C

by Ken Smolek

When I decided to do this article, it wasn't because I consider myself an expert in sorting techniques; on the contrary, it offered an opportunity to brush up in an area I haven't used in several years. I figured, what the heck, I'll dig up one of the myriad books on sorting, whip together a few representative examples in C, and have the whole thing done in three or four hours. Unfortunately, reality intruded. Different people have different 'quanta' of useful units of time: some can do useful work in a five-minute interval, while others need 15 minutes to get anything done. I need at least half an hour, and an hour is more realistic. In addition, the task itself grew somewhat, as they have a tendency to do. So, many hours later, I've done more than originally intended, less than ultimately desired, and hopefully something useful.

Most books which discuss sorting use integers rather than characters as their basic unit of data. This has the advantage of simplifying the overall problem, so that the process doesn't get lost in the implementation. In keeping with this tradition, I will give three examples of integer sorts. If you're like me, however, you seldom have occasion to sort a set of integer values. Therefore, a fourth sort program is given here which uses the Quicksort algorithm to sort an array of fixed-length records, with an unsigned character key of an arbitrary length in an arbitrary position in the record. The sort can be in ascending or descending order.

Some dBase II files were used for debugging the sorts. This simplified testing by providing large sets of data after conversion to their SDF format.

A few objectives, limitations, and disclaimers should be mentioned:

* The three sort types implemented are insertion sort, Shellsort, and Quicksort, because they are easy to understand and commonly used.

* I have attempted to keep the code as clean and as portable as possible. The three integer sorts are very straightforward, using arrays to clarify the operations taking place.

* The full-blown record-array Quicksort accepts command line arguments, reads an input file, and uses pointers, and so is somewhat harder to understand.

* These are examples of sorting in C which attempt to display both the algorithms and some of the power of the C language. There is no claim that these implementations are in any sense optimal.

* Compilation was done with standard Lattice C, but there should be no problem compiling under any major C compiler. (I had planned to test with both C86 from Computer Innovations and DeSmet C from C Ware, but time ran out. Maybe next time.)

* These are all internal sorts, so the entire array of records being sorted must be in memory at the same time. (Methods of external sorting may be discussed in a future article.)

The type of sort you choose should depend upon factors such as the number and size of the records typically being sorted, the relative importance of speed, and the number of times the sort will be used. Insertion sort is a good introduction to sorting because it is intuitive and works well for fairly small numbers of elements. Shellsort is a good general-purpose sort, particularly if you are in a hurry, because it is fairly trivial to implement and is at least half as fast as Quicksort even for large numbers of elements. Quicksort is commonly considered to be the best general-purpose sorting method, because it is fairly simple and is fast for all values of N (the number of elements).

INSERTION SORT

This very simple sorting algorithm operates by inserting each element into its proper place among elements already considered. An element is inserted into proper position by moving all larger elements one slot to the right and inserting the element into the slot vacated.

SHELLSORT

Shellsort extends the basic insertion sort by moving elements more than one position at a time in the insertion process. In this way, elements which are far from their final position move much more quickly to their final destination. This is done by choosing a series of intervals, starting with a large value and terminating with the value equal to one. At each stage, an insertion sort is performed on elements which are an "interval" apart. The final stage, with the interval equal to one, is an

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ordinary insertion sort, which leaves the array in fully sorted order. The set of intervals is somewhat arbitrary.

An empirical set of the form

interval = 3*interval + 1

works well and is used here. The first interval chosen is always the largest interval smaller than N.

QUICKSORT

Quicksort is the inspired brainchild of C.A.R. Hoare, who first published it in 1960. It improves performance by working from both ends of the array and by dividing the array up into smaller arrays. An arbitrary partitioning element is chosen, and array elements are compared to this value. Two pointers are used; one moves forward from the beginning of the array, while the other moves backward from the end of the array. The first pointer stops whenever it encounters an element larger than the partitioning value, while the second pointer stops whenever it encounters a smaller one. The two elements are then interchanged. When the pointers cross, the process is complete: all elements smaller than the partitioning element are to the left of the crossing point, while all elements larger are to the right. The crossing point is then used as the divider between two new partitions, which are sorted by the same method. When all partitions are only one element long, the sort is complete. The basic algorithm is shown clearly in the integer Quicksort listing.

Quicksort, which works best in a randomly organized file, has the disadvantage of working very slowly on files which are already sorted because of the partitioning process. The disadvantage can be avoided by taking three elements from the partition (typically the left, center, and right) and averaging them to create the partitioning value used. In addition, recursion can be eliminated by using an explicit stack, which provides a further increase in speed. A third improvement is to use a Quicksort call only for partitions larger than a certain number of elements (say 10 or 20) and use insertion sort for all smaller partitions. These are bells and whistles, and have not been implemented in the examples given.

COMPARISONS

The Integer Sorts:

The three integer sort examples are not designed for optimal performance, but only to show the processes involved. A 20-integer array, which is compiled into the program, is copied to a secondary array and sorted. For timing purposes, the entire process is repeated 10,000 times. Several printf statements are included (but commented out) to allow an examination of the step-by-step process. On my system (an 8-Mhz 8086-based MS-DOS machine), the following times were obtained:

Insertion sort	41.5 seconds
Shellsort	44.3 seconds
Quicksort	41.0 seconds

Not much difference there; it would show up more dramatically if a larger array were sorted.

The Record-Sorting Quicksort Program:

I implemented the record-sorting version for all three sort types; only the Quicksort version is shown here. As a test, a file of 357 80-byte records, with a 4-byte key, was sorted by all three methods, with the following results:

Insertion sort	69.0 seconds
Shellsort	11.3 seconds
Quicksort	6.3 seconds

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The record sort accepts 5 arguments on the command line, in the format

quiksort -rnn -knn -lnn [-d] infile outfile

The arguments specify the following:

-rnn	nn specifies the record length in bytes (including any record separators)
-knn	nn specifies the key position in the record (0 means the beginning of the record)
-lnn	nn specifies the key length in bytes
-d	sort in descending order (optional parameter)
infile	the input file to be sorted
outfile	the output file where the sorted file is written

The entire key must lie within the record (i.e., it is not valid to specify a record length of 40, a key position of 35, and a key length of 10). The "-" arguments may be specified in any order. The "d" argument may be left out. Any existing outfile is destroyed if no outfile is specified. In this instance the outfile is written to the infile, destroying the original. (It is recommended that you always write to a second file.)

If your file for sorting was called "testfile," the command line might look like this:

quiksort -r79 -k0 -l4 testfile outfile

In the full Quicksort version, I tried to give examples of a number of different features of the C language, along with error and command line processing. Any comments you have would be appreciated.

A copy of the sorts on diskette can be obtained from

Ken Smolek
NW Microsystem Design
PO Box 10853
Eugene, OR 97401
ph. 503-484-7129

Include $5 to cover media and postage.

Ad Changed 1/20/84

PROGRAMMERS- COMPARE Products. Get Answers. GUARANTEES

We research and specialize only in software for programmers of micros. If a product is unknown to you and we recommend it, then we take the risk.

PHILOSOPHY: We carry every programmer's product for CPM80, MSDOS and CPM86 plus every key product for APPLE, Commodore 64, ATARI and TRSDOS.

"C" LANGUAGE

			LIST PRICE	OUR PRICE
APPLE:	AZTEC C-Full, ASM		$199	call
8080:	BDS C-Fast, popular		150	125
8080:	AZTEC C-Full		199	call
Z80:	ECOSOFT-Fast, Full		250	225
8086:	C86- optimizer, Meg		399	call
8086	Lattice - New1.1& 2.0		600	495
MicroSoft (Lattice)		MSDOS	500	call
Digital Research - Megabyte		8086	350	280
Desmet by CWare Fast		8086	199	99

BASIC

	ENVIRONMENT		
Active Trace-debug	8080/86	$ 80	73
MBASIC-80 - MicroSoft	8080	375	$255
BASCOM-86 - MicroSoft	8086	395	279
CB-86 - DRI	CPM86/PC	600	449
PBASIC - DRI	8086	150	119
Business BASIC-MicroS	MSDOS	600	449

EDITORS Programming

BELLESOFT - PASCAL	MSDOS	$ 245	call
C Screen Editor - source	8080/86	NA	60
EDIX	PCDOS	195	149
FINAL WORD	8080/86	300	225
MINCE	CPM, MSDOS	175	149
PMATE - powerful	CPM	195	175
	8086	225	195
VEDIT - full, liked	CPM, PCDOS	150	119
	8086	200	159

PASCAL

	ENVIRONMENT	LIST PRICE	OUR PRICE
PASCAL MT+86	CPM86/PC	$400	$295
without SPP	CPM80	350	259
MS PASCAL 86	MSDOS	350	280
ALCOR PASCAL full w/ right extensions to learn	8080	* 250	call
SBB PASCAL great fast	PCDOS	350	315
PASCAL 64 - nearly full	COM 64	99	89
UCSD PASCAL Compiler	UCSD-PC s	375	call

LANGUAGE LIBRARIES

C_to_dBASE interface	8086/86	NA	$95
C TOOLS Graph Str	PCDOS	NA	115
FLOAT87 Lattice Pt	PCDOS	NA	115
GRAPHGSX 80	CPM80	60	50
HALO	PCDOS	150	125
ISAM Access Manager 86	8086	400	300
PHACT with C	PCDOS	NA	250
FABS	CPM80	150	135
PASCAL TOOLS	PCDOS	NA	115
SCREEN Display Mgr 86	8086	500	375
PANEL 86 many	PCDOS	NA	315
WINDOWS - AMBER	PCDOS	295	call

FORTRAN

MS FORTRAN-86 - Megabyte	MSDOS	$350	$265
SS FORTRAN-86	CPM-86	425	345
FORTRAN-80 - 66 decent	CPM80	500	350
INTEL FORTRAN-86	IBM PC	NA	1400
DR FORTRAN COMING			
RM FORTRAN COMING			

OTHER PRODUCTS

Microshell - improve CPM	8080	150	125
IQ LISP - full, 1000K RAM	PCDOS	175	call
PLINK-86 - overlays	8086	350	315
PL/1-86	8086	750	560
Prolog - AI Language	PCDOS	NA	125
Lookalike of Haves Smart 1200 modem		369	

LOOKING FOR

- *FORTRAN to C conversion software
- *AI Library for C
- *Benchmark info on PASCAL, FTN, C
- *Faster, compatible MSDOS Assembler
- *Full Interpreter for C

PRODUCTIVITY USING MSDOS

Assume use of a compiler and typical editor. What commercial or public domain products improve productivity? "MSDOS Programming" is a growing document with some answers. Call to request it. **Help improve it.** **Earn $50** credit toward any purchase when we add to "MSDOS Programming" any description, code, or idea received from you in the mail or on our BBS. Include your name, phone, compiler you use.

SUGGESTED PRODUCTS FOR PRODUCTIVITY

- Full use of files across 2 0 directories **AKA ALIAS** $60. Overlays, headers in alt directory.
- Recover erased or damaged files - use **Norton Utilities** $65 or **POWER** $139 which also includes directory management, memory move, bad sectors file, etc.
- Programming tools for filecompare, cross reference, combining/splitting files, linenumber, radix conversion, columnmove, scrollspeed, prettyprint use **Programmer's Helper** $175.
- Improve on DEBUG with **Codesmith-86** $135 many breakpoints/windows. Disasm, save to disk etc.
- Control/run multiple tasks simultaneously. And/or RAMdisk, better Printspool **Multijob** $179

SERVICES

- * Programmer's Referral List
- * Compare Products
- * Help find a Publisher
- * Evaluation Literature
- * Dealer's Inquire
- * Newletter
- * Rush Orders

* Bulletin Board-7 PM to 7 AM 617-461-0174

Call for a catalog, literature, comparisons, prices. Shipping $2.50 per item purchased. All FORMATS available.

THE PROGRAMMER'S SHOP™

908-J Providence Highway, Dedham, MA 02026

617-461-0120, Mass: 800-442-8070

VISA 800-421-8006 MASTER CARD

RECENT DISCOVERIES HIGHLIGHTED

C Helper includes source in C for CP/M80, MSDOS for a: DIFF, GREP, Flowcharter, XREF, C Beautifier and others, $115.

Note: all prices subject to change without notice.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
/*
        insort: insertion sort

        by Ken Smolek
        created         12/18/83, 2015 hrs
        last modified   12/18/83, 2130 hrs

        Simplified insertion sort; works on an integer array

*/

#include        "stdio.h"

/*      array of integers for test      */

int     irecs[] = {11,1,13,19,12,3,20,5,14,8,2,17,10,16,4,18,9,7,6,15};
int     recs[20];

main()
        {int    iter,i;
        printf("10,000 iterations\n");
        for (iter = 0; iter < 10000; iter++)
            {for (i = 0; i < 20; i++)
                recs[i] = irecs[i];
/*          prtarray(recs,20);  */
            insort(recs,20);
            }
        prtarray(recs,20);
        }


insort(recs,nrecs)
        int     recs[],nrecs;
        {int    kval,i,j;
        for (i = 1; i < nrecs; i++)
            {kval = recs[i];
            for (j = i; (j > 0)&&(recs[j-1] > kval); j--)
                recs[j] = recs[j-1];
            recs[j] = kval;
/*          prtarray(recs,i);     */
            }
        }

prtarray(a,n)
        int     a[],n;
        {int    i;
        for (i = 0; i < n; i++)
            printf("%d ",a[i]);
        printf("\n");
        }
```

```
/*
        ssort: shell sort

        by Ken Smolek
        created         12/18/83, 2015 hrs
        last modified   12/31/83, 1230 hrs

        Simplified insertion sort; works on an integer array

*/

#include        "stdio.h"

/*      array of integers for test      */

int     irecs[] = {11,1,13,19,12,3,20,5,14,8,2,17,10,16,4,18,9,7,6,15};
int     recs[20];

main()
        {int    iter,i;
        printf("10,000 iterations\n");
        for (iter = 0; iter < 10000; iter++)
            {for (i = 0; i < 20; i++)
                recs[i] = irecs[i];
/*          prtarray(recs,20);              */
            sort(recs,20);
/*          prtarray(recs,20);              */
            }
        prtarray(recs,20);
        }


sort(recs,nrecs)
        int     recs[],nrecs;
        {int    interval,kval,i,j;
        for (interval = 1; interval <= nrecs; interval = 3*interval+1)
            ;
        do
            {interval /= 3;
            for (i = interval; i < nrecs; i++)
                {kval = recs[i];
                for (j = i; (j >= interval)&&(recs[j-interval] > kval);
                  j -= interval)
                    recs[j] = recs[j-interval];
                recs[j] = kval;
                }
/*          prtarray(recs, i);              */
            } while (interval > 1);
        }

prtarray(a,n)
        int     a[],n;
        {int    i;
        for (i = 0; i < n; i++)
            printf("%d ",a[i]);
        printf("\n");
        }
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PROGRAMMERS:
If You Own A Compiler You Need

POLYLIBRARIAN™

The
OBJECT MODULE LIBRARIAN
For MS-DOS (PC-DOS)

POLYLIBRARIAN lets you create a single library from related Object Modules. The IBM PC Linker will then select only the modules necessary to produce an executable file.

For added flexibility, POLYLIBRARIAN works with any compiler or assembler that uses the MS-DOS (IBM PC) Linker.

POLYLIBRARIAN lets you use your time more productively because you'll eliminate repetitious programming steps.

- Shrink Development Time
- Optimize Object Modules
- Unlock The Secrets of Existing Libraries

Three Operating Modes

The Interactive Mode, Command Line Mode, and Command File Mode allow first-time librarian users and advanced users to realize the full power and potential of POLYLIBRARIAN. The Fully Nested Command Modes have built-in help features.

Comprehensive User's Manual

The highly praised User's Manual explains the functions of a librarian, describes the use of each operating mode, includes a Command Summary, list of error messages, the Intel Object Module Format, and a complete BNF description of the POLYLIBRARIAN Grammer. A step-by-step tutorial lets you learn to create libraries, add and delete modules, extract modules and more.

Requires 128K RAM MS-DOS 1.0, 1.1 or 2.0

VISA/MC ORDERS CALL
1-800-547-4000
ask for Dept. 306

$99 Includes Manual
(VISA ADD $1.50)

OR SEND CHECK, P.O.'s TO:

POLYTRON™

DS-3-306
P.O. Box 787
Hillsboro, OR 97123
(503) 648-8595

```
/*
        qsort: insertion sort

        by Ken Smolek
        created         12/18/83, 2015 hrs
        last modified   12/31/83, 1115 hrs

        Simplified insertion sort; works on an integer array

*/

#include        "stdio.h"

/*      array of integers for test      */

int     level;
int     irecs[] = {11,1,13,19,12,3,20,5,14,8,2,17,10,16,4,18,9,7,6,15};
int     recs[20];

main()
        {int    iter,i;
        printf("10,000 iterations\n");
        for (iter = 0; iter < 10000; iter++)
            {for (i = 0; i < 20; i++)
                recs[i] = irecs[i];
/*          prtarray(recs,20);              */
            sort(recs,0,19);
/*          prtarray(recs,20);              */
            }
        prtarray(recs,20);
        }


sort(recs,first,last)
        int     recs[],first,last;
        {int    kval,tval,i,j;
/*      printf("%d %d\n",first,last);           */
        if (first >= last)
            return;
        printf("%d\n",level++);
        kval = recs[last];
        i = first-1;
        j = last;
/*      printf("%d %d\n",i,j);                                  */
        do
            {while ((recs[++i] < kval)&&(i < j))
/*              printf("i %d %d %d\n",i,recs[i],kval);          */
                ;
            while ((recs[--j] > kval)&&(j > i))
/*              printf("j %d %d %d\n",j,recs[j],kval);          */
                ;
            tval = recs[i];
            recs[i] = recs[j];
            recs[j] = tval;
/*          printf("%d %d\n",i,j);                              */
/*          prtarray(recs,20);                                  */
            } while (i < j);
        recs[j] = recs[i];
        recs[i] = kval;
        recs[last] = tval;
/*      prtarray(recs,20);                                      */
        sort(recs,first,i-1);
        sort(recs,i+1,last);
        printf("%d\n",--level);
        }

prtarray(a,n)
        int     a[],n;
        {int    i;
        for (i = 0; i < n; i++)
            printf("%d ",a[i]);
        printf("\n");
        }
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
/*
        qsort: quick sort

        by Ken Smolek
        created         12/18/83, 2015 hrs
        last modified   1/1/84, 1600 hrs

        Quick sort; works on an array of records
        with an unsigned character key

        Uses pointers

*/

#include "stdio.h"
#include "ctype.h"

#define LENFNAME 16                     /* length of filename buffers */

extern char *malloc();
extern long lseek();

/*      global storage                  */

char    infn[LENFNAME],outfn[LENFNAME];
char    *recs,*reccopy,*tmpcopy,*getargs();
int     reclen,keypos,keylen,descend,infd,outfd;
unsigned nrecs;


main(argc,argv)
        int     argc;
        char    *argv[];
        {char   *msgptr;
        int     i;
        unsigned rdcnt;
        if ((msgptr = getargs(argc,argv)) != (char *)NULL) /* get arguments */
            {printf(msgptr);            /* error in command line */
            exit(0);
            }
        recs = malloc(reclen);          /* allocate one record */
        for (nrecs = 0; (rdcnt = read(infd,recs,reclen)) == reclen; nrecs++)
            ;
        free(recs);
        recs = malloc(nrecs*reclen);    /* allocate record array */
        if (recs == (char *)NULL)
            {printf("not enough memory\n");
            exit(0);
            }
        lseek(infd,0L,0);
        for (i = 0; i < nrecs; i++)
            {rdcnt = read(infd,recs+i*reclen,reclen);
            if (rdcnt != reclen)
                {printf("error reading input file\n");
                free(recs);
                close(infd);
                close(outfd);
                exit(0);
                }
            }
        reccopy = malloc(reclen);               /* allocate work areas */
        tmpcopy = malloc(reclen);
        printf("starting sort\n");
        sort(0,nrecs-1);                        /* sort the array */
        printf("sort complete, writing file %s\n",outfn);
        free(reccopy);                          /* release work areas */
        free(tmpcopy);
        outfd = creat(outfn,1);                 /* open output file */
        if (outfd == -1)
            printf("can't open output file\n");
        else
            {for (i = 0; i < nrecs; i++)
                write(outfd,recs+i*reclen,reclen);
            }
        free(recs);
        close(infd);
        close(outfd);
        exit(0);
        }

/*
        getargs: process the command line

        the command line has the form

            sort -rnn -knn -lnn infile outfile

        where   -rnn specifies the record length
                -knn specifies the key position in the record
                -lnn specifies the key length
                -d   specifies descending order
                infile is the input file to be sorted
                outfile is the output file

        returns null pointer if the command line is good, a
        pointer to the error message if not
*/

char *getargs(argc,argv)
        int     argc;
        char    *argv[];
        {char   *argp;
        reclen = keypos = keylen = -1;  /* initialize arguments */
        descend = 0;                    /* assume ascending order */
        while (--argc)                  /* process all -x arguments */
            {argp = *++argv;            /* get ptr to argument */
            if (*argp++ != '-')         /* test for - argument */
                break;
            switch(toupper(*argp))      /* determine arg type */
                {case 'R':              /* record length */
                    reclen = atoi(++argp);
                    break;
                case 'K':               /* key position */
                    keypos = atoi(++argp);
                    break;
                case 'L':               /* key length */
                    keylen = atoi(++argp);
                    break;
                case 'D':               /* descending order sort */
                    descend = 1;
                    break;
                default:                /* illegal argument */
                    return("illegal command line argument\n");
                }
            }
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        if (reclen ( 0)                 /* test for valid record length */
            return("invalid record length\n");
        else if ((keypos ( 0)!!(keypos )= reclen))/* test key position */
            return("invalid key position\n");
        else if ((keylen (= 0)!!(keypos+keylen ) reclen))/* test key length */
            return("invalid key length\n");
        if (!argc--)                    /* should be more arguments */
            return("no input file specified\n");
        strcpy(infn,*argv);             /* save input filename */
        infd = open(infn,0);            /* open input file */
        if (infd == -1)                 /* test for error */
            return("can't open input file\n");
        if (!argc)                      /* test for writing to same file */
            strcpy(outfn,infn);         /* output to same file */
        else
            strcpy(outfn,*++argv);      /* output to different file */
        return (char *)NULL;            /* good command line */
        }

/*
        main sort module

        this is not a completely general sort module.  For speed,
        it assumes that the record array is in recs[], and that
        reclen, keypos, and keylen are global variables.  Since
        these are constants, it is a reasonable alternative to
        passing them as arguments.
*/

sort(first,last)
        int     first,last;
        {char   *iptr,*jptr;
        int     i,keyval;
        if (first )= last)
            return;
        copystr(reccopy,recs+last*reclen,reclen);
        i = first-1;
        iptr = recs+i*reclen;
        jptr = recs+last*reclen;
        do
            {while (1)
                {i++;
                iptr += reclen;
                keyval = keycmp(iptr+keypos,reccopy+keypos,keylen);
                if (descend)
                    keyval = -keyval;
                if ((keyval )= 0)!!(iptr )= jptr))
                    break;
                }
            while (1)
                {jptr -= reclen;
                keyval = keycmp(jptr+keypos,reccopy+keypos,keylen);
                if (descend)
                    keyval = -keyval;
                if ((keyval (= 0)!!(jptr (= iptr))
                    break;
                }
            copystr(tmpcopy,iptr,reclen);
            copystr(iptr,jptr,reclen);
            copystr(jptr,tmpcopy,reclen);
            } while (iptr ( jptr);
        copystr(jptr,iptr,reclen);
        copystr(iptr,reccopy,reclen);
        copystr(recs+last*reclen,tmpcopy,reclen);
        sort(first,i-1);
        sort(i+1,last);
        }
```

```
/*
        copystr: copy a string of length n
*/

copystr(d,s,n)
        char    *d,*s;
        unsigned n;
        {while (n--)
            *d++ = *s++;
        }

/*
        keycmp: compare two unsigned character strings of length n

        returns         = 0 if strings are equal
                        ) 0 if string1 ) string2
                        ( 0 if string1 ( string2
*/

keycmp(s1,s2,n)
        char    *s1,*s2;
        int     n;
        {for ( ; *s1 == *s2; s1++, s2++)
            {if (!n--)
                return 0;
            }
        return ((*s1&0xff) - (*s2&0xff));
        }

/*
        atoi: convert an ASCII string to an integer value

        (this function is required because Lattice C does not
        provide the standard C function atoi)
*/

atoi(s)
        char    *s;
        {int    val;
        val = 0;
        stcd_i(s,&val);
        return val;
        }

/*
        prtrec: print a string of length n
*/

prtrec(p,n)
        char    *p;
        int     n;
        {while (n--)
            putchar(*p++);
        putchar('\n');
        }
```

□

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PC WRITE AND QUICKSOFT

PC WRITE is a word processor offered under the Shareware system from Quicksoft. You may acquire a free copy of PC WRITE from a friend or from Quicksoft for a nominal $10. If you decide to become a licensed owner ($75) then you also receive ongoing updates, $25 commissions, and A COMPLETE COPY OF THE PASCAL AND ASSEMBLY SOURCE CODE totalling over 200K. Bob Wallace, the author of PC WRITE, did the major design work on the Microsoft Pascal Compiler. We will talk briefly with Bob, then take a look at PC WRITE.

PJ: When we opened our copy of PC WRITE we were quite excited to see that the source code was included.

BOB: Oh sure. Since the word processor is free I figured that nobody would be able to undercut our price.

PJ: A Pascal or assembly programmer would reap a tremendous benefit from going over your programs.

BOB: I do know Pascal pretty well, and the assembly modules could be used independently in other Pascal programs. I have not tried to do any marketing of PC WRITE as a software writer's tool, all our time is spent promoting it as a word processor.

PJ: Could we reprint a piece of your Pascal and assembly code for our subscribers so they can see what they would be getting?

BOB: Sure. As far as the assembly code goes, some of the modules from the EB.ASM file would be most applicable to other Pascal programs. Most of the low level tools for PC WRITE are written in assembly. PR.PAS may have some file routines that would be representative.

PJ: When did you leave Microsoft?

BOB: January, about a year ago.

PJ: What was your relationship to the Pascal Compiler?

BOB: I designed the language (the extension part of it), wrote the front end, and coded a fair amount of the run time, about half of it.

PJ: The Pascal compiler creates a large stack and hauls in a bunch of run time modules. Is there any way to cut down on the size?

BOB: The editor (PC WRITE) does not use any of the Pascal run time. The EB.ASM module substitutes for most of it. If you do something complex that cannot be handled with in-line Pascal code then you tend to call in the runtime modules, anything that uses SETS or LSTRINGS or I/O. I just don't use SETS and LSTRINGS, so I do not need the run time part. I don't use real numbers either.

The compiler is selective and the library is broken down into smaller modules. You COULD break it down even further into individual subroutines. The way the Linker and the Library are set up every module must be at least 512 bytes. The modules on disk are sectioned into 512K pieces but this is not required when you load it into memory. The size of the run time and libraries reflects a desire to be safe. When an error occurs you want to give an error message; this requires an error checking module calling a message module that calls a routine to find out what line number you are on and what the stack is doing. Consequently you need the whole I/O subsystem. It has a machine-independent and a machine-specific part.

The stack grabs all the memory it can get. It is pretty easy to cut this down. When you link you can say that you want to allocate DATA memory from DGROUP. You can say you want to allocate it from zero memory upward or from FFFF downward. We like to allocate it downward. When the program starts, there is a bunch of allocated memory which will end up being at the top of the DATA segment. Immediately below that is code. If you looked at DS:0000 it would be down in the code somewhere. Or sometimes even below the bottom of the address space. If your total code net is less than 64K then DS:0000 is really below the bottom, but then there is no stack and no heap. So when we get control in PC WRITE, we grab more memory above, however much we can find up to 64K, and we shift everything upwards with a block move instruction. Then we add to DS by the same amount. The data is still in the same place in relation to DS. We increased DS and shifted the data upward. This is all done in the entry point routine.

PJ: Is this a special part of PC WRITE code?

BOB: An entry routine is provided in source code with the Pascal compiler. It is possible, without very much tinkering, to avoid shifting up quite so much.

PJ: We received a card from you referring to a COMDEX convention bash that mentioned Silicon Northwest. The idea is great. Exactly what is it?

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BOB: Silicon NW is a first attempt at formal cooperation among local independent software firms. So far it is mainly for convention gatherings, to share the costs and have a location at faires where the press and other industry members can meet us. There is another group here in the Northwest that I call the Micro Computer Marketing Forum. We meet once a month here in Seattle. We usually have around 40 to 50 people attend. We have a talk and then kind of a random access networking period. It is mostly a place for people to meet one another - find a marketing manager, printing outlets, etc. No newsletter yet. I keep the mailing list here at Quicksoft. Anyone can call here if they would like to know more.

PJ: What projects are you currently working on?

BOB: I am writing some enhancements to the editor: horizontal scrolling, better printer support, and keyboard macros.

PC WRITE - WORD PROCESSOR

PC WRITE gives you on-line HELP, commands are keystroke driven, control keys can be configured, display colors can be altered, any ASCII character can be inserted into the text, and it has un-delete/un-replace keys. The editing functions include split screen capabilities, word wrap, margin ruler, and paragraph justification. Page formatting options are provided, as well as footers and extra binding margins. The editor is very fast.

PC WRITE is an excellent product. To order a copy, register or find out more information contact:

Quicksoft
219 First N. #224
Seattle, WA 98109
(206-282-0452)

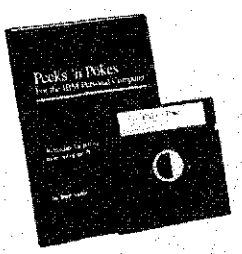
The following listings show a directory of the source code diskette, and a portion of the Pascal and assembly programs sent to registered owners of PC WRITE.

```
EB          INT      2430   12-10-83    2:43a
ED          PAS     43812   12-10-83    2:44a
EC          INC     37400   12-10-83    2:44a
PR          PAS     17610   12-10-83    2:44a
DB          PAS      1231   12-10-83    2:44a
EA          ASM     55510   12-10-83    2:44a
EB          ASM     22898   12-10-83    2:44a
EE          ASM      9406   12-10-83    2:44a
EXPAND      COM       800   11-01-83    1:24p
COMPRESS    COM       892    9-05-83   11:42p
README      SRC      1276   11-02-83    6:11p
EXPAND      ASM     14892   12-10-83    2:44a
EA          INT      1846   12-10-83    2:43a
```

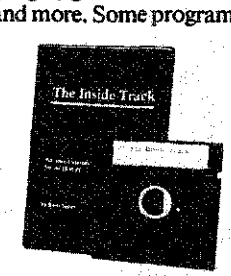

Know Thy PC!

Are you writing programs in BASIC or Pascal? The popular **Peeks 'n Pokes** has a disk with 58 programs and a 38-page manual that helps you get 'underneath the covers' of the PC. Learn how to use PEEK, POKE, INP, OUT, and DOS/BIOS function calls to do what you want, fast! Do you want to perform functions not available from BASIC or Pascal? It's all explained in the manual and demonstrated in the sample programs. Source code included!

Peeks 'n Pokes shows you how to:
- Access the system's configuration
- Unprotect BASIC programs
- Scroll part or all of the screen
- Access the file directory
- Logically swap printers
- Read and change the keyboard
- Find more Peeks and Pokes
- And much more... for only $30.00



Want to know more? **The Inside Track!** is a collection of advanced utilities for the PC programmer. It contains a disk with 61 programs, a 42-page manual, and a fold-out memory map that helps you get better performance from the PC. With this package you can give your programs assembler-assisted speed from high-level languages, get control over memory, customize and control the PC, and more. Some programs require DOS 2.00. Source code included!



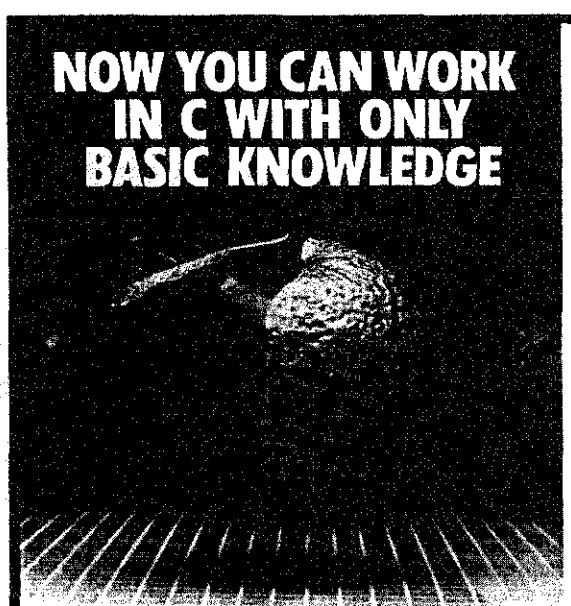
The Inside Track! shows you how to:
- Read/write files as fast as DOS
- Display data on the screen faster
- Reserve memory for your use
- Copy memory to another location
- Copy-protect your programs
- Load large programs faster
- Control the keyboard settings
- And much more... for only $45.00

MasterCard and VISA accepted. Shipping charges: $2.50 per order for UPS; $2.50 per item for First Class Mail to USA and Canada; $6.00 per item for Air Mail outside USA and Canada. Dealer inquiries invited.

Data Base Decisions • 14 Bonnie Lane • Atlanta, GA 30328 • 404/256-3860

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
INTERFACE;  UNIT ebunit (* General routines implemented in assembly *)

(keyflag, keyscan, keycode, numlock, scrlock, shftkey, ctrlkey, shftset,
 ctrlset, keytype, putfile, getfile, dosfile, dosopen, dosclos, dosgets,
 dosputs, mover, movel, beep, begmqq, endmqq, cesxqq);

{keytype returns true when a key has been typed, and sets keyflag to the
 shift bits, keyscan to the scan returned, and keycode to the code returned;
 it also maintains numlock and scrlock based on keyflag, sets shftkey/ctrlkey
 based on shift and control keys, and forces them based on shftset/ctrlset}
VAR  keyflag, keyscan, keycode: byte;
     numlock, scrlock, shftkey, ctrlkey, shftset, ctrlset: boolean;
FUNCTION keytype: boolean;

{PUTFILE creates and writes a DOS file with a given name from a memory block;
 starting at BEG and ending at MAX, returning first unwritten offset in ENF.It
 returns false if any I/O error occurs, except if disk full w/ ENF < MAX}
FUNCTION  putfile (VAR enf:adrmem; beg,max:adrmem; CONST name:string): boolean;

{GETFILE opens and reads a DOS file with the given name to a block in memory,
 starting at BEG and ending at MAX, returning the first unused offset in ENF.
 It returns false if any I/O error occurs, particularly file-not-found.}
FUNCTION getfile (VAR enf:adrmem; beg,max: adrmem; CONST name:string): boolean;

{These primitives open, close, read, or write a file.  They do either DOS 1.xx
 FCB calls, or DOS 2.xx file handle calls.  All return false on any error.}
TYPE dosfile = string(40);
{open a file with the name;  if new, then create it, else use existing}
FUNCTION dosopen (var f: dosfile; new: boolean; const name: string): boolean;
FUNCTION dosclos (var f: dosfile): boolean; {close the file}
{read/write from BEG to MAX;  read sets ENB to the first unused buffer byte}
FUNCTION dosgets (var f: dosfile; var enb: adrmem; beg, max: adrmem): boolean;
FUNCTION dosputs (var f: dosfile; beg, max: adrmem): boolean;

{bytestream move routines;  both move length bytes from source to target}
PROCEDURE mover (source, target: adrmem; length: word);  {start at right end}
PROCEDURE movel (source, target: adrmem; length: word);  {start at left end}

PROCEDURE beep;                 {short toot on the built-in speaker}

VAR begmqq, endmqq: adrmem;     {beginning, end of available memory}
    cesxqq: word;               {segment value for program header}

END;
```

Enjoy the Best of Both Worlds with BASIC_C™!

Write or Call Collect

SOURCE
C Source
12801 Frost Rd.
Kansas City, MO 64138
816-353-8808

Until now, IBM BASIC and C were about as far apart as apples and oranges — and if you wanted to move from BASIC to C, you had to go back to square one. Not anymore!

There's no denying that BASIC is a fine language: it has probably helped you learn a lot about programming computers. But now that you know how to program, isn't it time to take the next step — to a language that was designed to help you quickly develop software that is fast, compact, and portable?

BASIC_C can help you make the transition smoothly: it's a library of C functions providing BASIC's high level capabilities you've come to know so well. Functions for string handling, sequential and random i/o, graphics, and many more — all designed to look and work like BASIC.

You get the source code to BASIC_C: extensively documented, it provides many examples to help you learn how to write C functions, and if a function doesn't happen to suit your specific needs . . . change it!

C Source provides a complete development package: the BASIC_C Library, a full screen editor, and a choice of three C compilers. Buy it all, or just the pieces you need.

BASIC_C Library - source code	$175
EC - full screen editor	$ 85
BASIC_C & EC	$225
BASIC_C & DeSmet C Compiler	$275
BASIC_C & Computer Innovations C compiler	$475
BASIC_C & Lattice C compiler	$550
BASIC_C, DeSmet C Compiler, & EC	$325
BASIC_C, Computer Innovations C compiler, & EC	$525
BASIC_C, Lattice C compiler, & EC	$600

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
name    ebsm
page    60,132

;General purpose assembly routines for Pascal and PC-Write programs

;(C) Copyright 1982 by Quicksoft; All Rights Reserved

;Authors note:
;This source is copyrighted, including "derivations" as defined by law.
;If you want to use it or a derivation commercially, please call so we
;can negotiate a flat fee or royalty agreement.  Questions about this
;source are answered in a general way as part of your support service;
;more extensive answers or special modifications may be available on a
;consulting basis.  Using modified pc-write source voids your warranty.
;Please keep this copyright rights and royalties notice in all sources.
; - Thank you,  Bob Wallace.
;Quicksoft, 219 First Ave N. #224, Seattle, WA. 98109;  (206) 282-0452.

;Change history:
; 1.3:  Fixed bug so putfile works with dos 1.0 as well as 1.1.
; 1.4:  Removed indirect far calls for dos file routines.
; 1.4:  Added prtstr procedure for debugging support.

;Summary of provided services:

;  begxqq, etc.........trimmed version of pascal runtime entry
;  keytype, key vars...gets character and shift flags from keyboard
;  mover, movel........local data segment moves
;  getfile, putfile....whole file read/write routines
;  beep................beeps the speaker

; Microsoft MS-DOS runtime system control source
; [original version by Bob Wallace at Microsoft]
; trimmed version for PC-Write, Bob Wallace 3/83

;memory layout:
;hi -> command (may be overlayed)
;            himem   segment  class himem (always empty)
;      dgroup, commqq segment  class common
;      dgroup, const  segment  class const
;      dgroup, comads segment  class comads
;      dgroup, data   segment  class data
;      dgroup, stack  segment  class stack
;      dgroup, memory segment  class memory
;      dgroup, heap   segment  class memory
;            code    segments
;lo -> dos code and data (fixed)
;
;The linker is told to load low and use ds allocation.  Only 256 bytes
;of initial stack are allocated, and no heap at all.  Begxqq moves all
;dgroup to high memory, making a gap in which the stack grows downward.
;The editor uses the bottom portion of this gap for its text buffer.
;All normal Pascal segments are defined.  The editor ignores the heap.

extrn   entgqq:far              ;main program entry point

;first dummy code segment tells linker to load code lowest
;
inixqq  segment 'CODE'
inixqq  ends

;a few dummy segments for compatibility
;
heap    segment public 'MEMORY';always empty
heap    ends
memory  segment public 'MEMORY';always empty
memory  ends

;stack segment definition (fixed initial minimal length)
;
stack   segment stack  'STACK'
memlo   equ     this byte       ;lowest data byte address
        db      1024 dup (?)    ;1k stack should be plenty
sktop   equ     this byte
stack   ends

;system resident public data
;
data    segment public 'DATA'
public  begmqq,endmqq           ;bounds of area available for text
begmqq  dw      0
endmqq  dw      0
public  stkbqq                  ;stack start, to fix long goto
stkbqq  dw      0
public  cesxqq                  ;dos saved es value (command line)
dosoff  dw      0
cesxqq  dw      0
public  beghqq                  ;first header word in heap
beghqq  dw      0
public  curhqq                  ;pointer to current heap item
curhqq  dw      0
public  endhqq                  ;just past end of the heap
endhqq  dw      0
public  stkhqq                  ;stack limit, to check overflow
stkhqq  dw      0
data    ends

;other dummy segments for compatibility
;
comads  segment public 'COMADS' ;fortran named common ads's
comads  ends
const   segment public 'CONST'  ;constants (not empty)
const   ends
commqq  segment public 'COMMON' ;fortran blank common
commqq  ends
himem   segment public 'HIMEM'  ;high memory dummy segment
himem   ends

dgroup  group   data,stack,const,heap,memory,comads,commqq

;code for this module
;
entxqq  segment 'CODE'
        assume  cs:entxqq,ds:dgroup,es:dgroup,ss:dgroup

public  begxqq,endxqq,inifqq    ;main entry and exit points

;begxqq:  initialization code
;- move dgroup up as much as possible to get gap
;- set initial stackpointer, framepointer
;- call  entgqq (main program entry)
;
begxqq  proc    far
        mov     dx,offset dgroup:memlo ;ds offset to lowest byte
        shr     dx,1            ;make into word offset address
        mov     cx,32768        ;highest word address possible
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        sub     cx,dx           ;cx is count of words in dgroup segment
        shr     dx,1            ;make word
        shr     dx,1            ;  count into
        shr     dx,1            ;    paragraph count
        inc     dx              ;round; number of para's could use
        mov     ax,es:2         ;dos end paragraph (first segment not free)
        sub     ax,himem        ;this is number of para's available total
;       jb      memerr          ;(should not happen)
        xor     bx,bx           ;assume all para's in dgroup can be used
        sub     ax,dx           ;if positive, more available than can use
        jae     mema            ;if negative, minus number of para's unused
        sub     bx,ax           ;bx is number of para's unused in dgroup
mema:   sub     dx,bx           ;dx is number of para's to move dgroup
;
;move all of dgroup higher in memory, making room for stack and text
        mov     ax,dgroup       ;get assumed dgroup segment value
        mov     ds,ax           ;this is the old, source segment
        mov     cesxqq,es       ;save incomming es value in ds
        add     dx,ax           ;old segment plus para's used
        mov     es,dx           ;makes new, target segment
        mov     si,65534        ;source offset
        mov     di,si           ;target offset
        std                     ;set direction flag
        rep     movsw           ;move ds:si-- to es:di-- until cx-=0
        cld                     ;leave direction clear
        mov     ds,dx           ;final ds value (may be negative)
;
;initialize stack segment and pointer, framepointer, memory pointers
        mov     ss,dx           ;initialize stack segment  !interrupts off!
        mov     sp,offset dgroup:sktop  ;set stackpointer  ! this sequence!
        mov     stkbqq,sp       ;to reset main bp after long goto
        sub     stkbqq,6        ;must adjust so main frame address
        sub     bp,bp           ;initial frame pointer zero
        shl     bx,1            ;make count of
        shl     bx,1            ;  unused paragraphs
        shl     bx,1            ;    into ds offset to
        shl     bx,1            ;      first data byte
        mov     begmqq,bx       ;start of memory ds offset
        mov     ax,sp           ;start of stack
        sub     ax,1024         ;permit 1k for stack
        mov     endmqq,ax       ;end of stack, end of text
;the heap isn't used by PC-Write, but code's cheap...
        mov     beghqq,bx       ;start of heap ds offset
        mov     curhqq,bx       ;current heap item ds offset
        mov     word ptr[bx],1  ;current header; free
        add     bx,2            ;byte after end of heap
        mov     endhqq,bx       ;ds offset after end of heap
        add     bx,384          ;comfortable boundary
        mov     stkhqq,bx       ;stack overflow ds offset
        sub     bp,bp           ;initial frame pointer zero
;
;call main program, exit to dos when it returns
        call    entgqq          ;call main program
endxqq  label   far             ;termination entry point
        mov     dosoff,0        ;make sure jump offset zero
        jmp     dword ptr dosoff;return to dos
begxqq  endp

;in case someone uses this with $stackck+ or the heap unit
;[special note:  has not been tested yet with heap unit !]
soveqq  proc    far
ghieqq  label   far
ghseqq  label   far
ghveqq  label   far
        mov     ax,cs           ;error message
        mov     ds,ax           ;segment
        mov     dx,cs:memess    ;error message offset
        mov     ah,9            ;command, send message
        int     33              ;tell user out of memory
        jmp     endxqq
memess  dw      memesg
memesg  db      'Out of memory$'
        db      13,10
soveqq  endp

;support for debugging module; send message to console
public  prtstr
prtstr  proc    far
        pop     di
        pop     es
        pop     dx              ;adr of message # '$'
        mov     ah,9            ;command, send message
        int     33              ;message to consol=
        push    es
        push    di
        ret
prtstr  endp

;satisfy Pascal generated call
inifqq  proc    far
        ret
inifqq  endp

entxqq  ends

data    segment public 'DATA'
;values returned by keytype function
flagads dd      00400017H       ;ads of bios shift byte
putmask dw      1               ;error return mask for putfile
public  keyflag,keycode,keyscan,numlock,scrlock,shftkey,shftset,ctrlkey,ctrlset
keyflag db      0               ;shift flags
keycode db      0               ;key's ascii value
keyscan db      0               ;key's scan code
numlock db      0               ;num lock status
scrlock db      0               ;scroll lock status
shftkey db      0               ;shift keys, status
shftset db      0               ;shift keys, force on
ctrlkey db      0               ;control key, status
ctrlset db      0               ;control key, force on
fcb     db      40 dup (?)      ;file control block
data    ends

ebsmsq  segment 'CODE'
        assume  cs:ebsmsq,ds:dgroup,es:dgroup,ss:dgroup

public  keytype,keyinit,keyterm
;set keyflag to shift byte, and if keyboard char available, read it and
; set keycode and keyscan to char values.  return true iff char avail.
keytype proc    far
        push    bp
;first get all the shift bits
        mov     ah,2            ;command to get shift bits
        int     22              ;keyboard i/o
        mov     keyflag,al      ;return shift bits
        les     bx,flagads      ;get shift flag address
        test    al,20H          ;is numlock still on ?
        jnz     keynumy         ;always on for bios
        xor     numlock,1       ;change numlock status
        or      byte ptr es:[bx],20H  ;stuff bit into bios
keynumy:test    al,10H          ;is scrlock still on ?
        jnz     keyscry         ;always on for bios
        xor     scrlock,1       ;change scrlock status
```

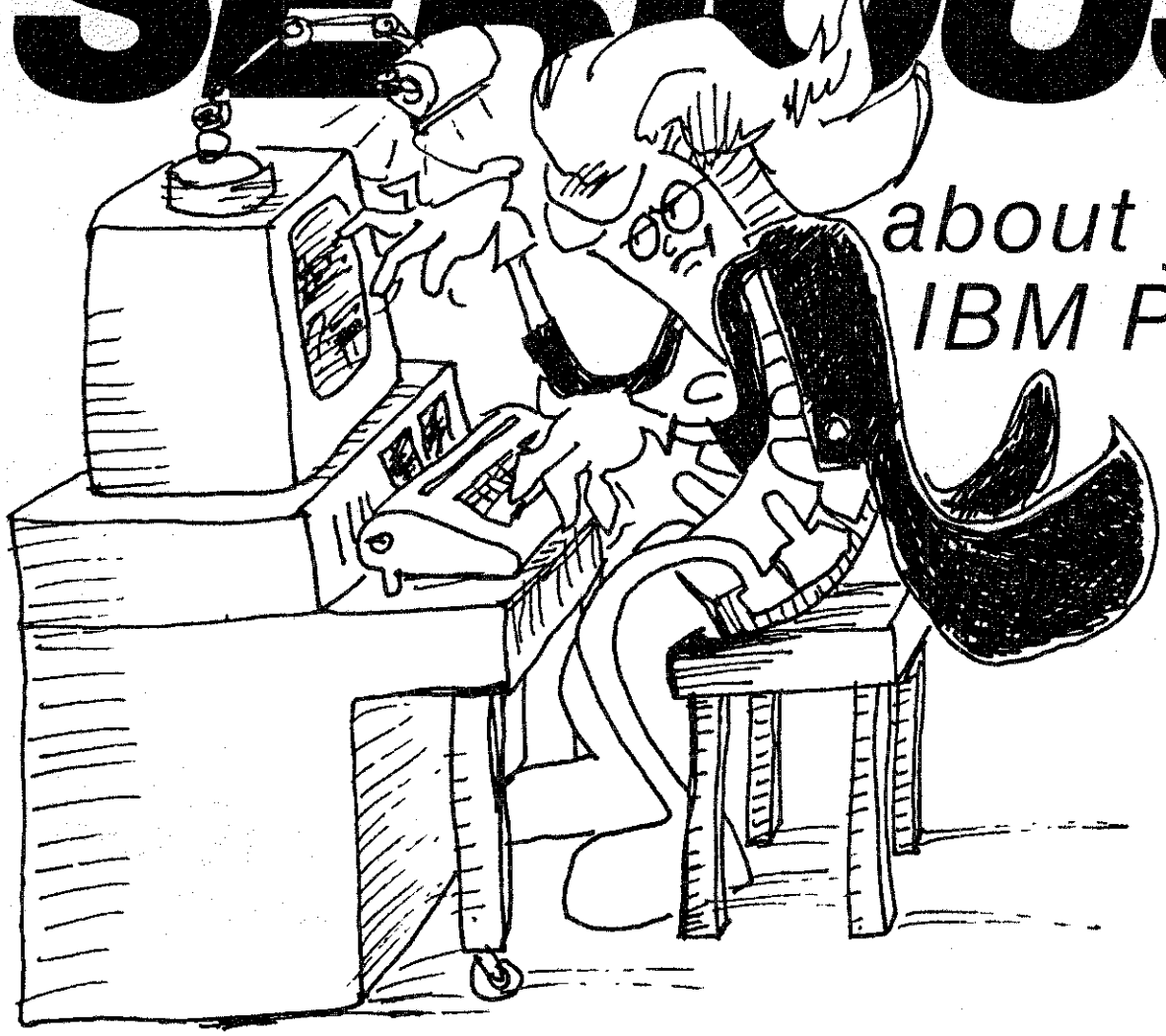

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        or      byte ptr es:[bx],10H  ;stuff bit into bios
keyscry:mov     ch,shftset      ;may want to force shift
        mov     cl,ctrlset      ;may want to force control
        shl     cl,1            ;position the bit
        shl     cl,1            ;position the bit
        or      cl,ch           ;merge shift and control
        or      byte ptr es:[bx],cl   ;set bits directly
        mov     ah,al           ;duplicate shift bits
        shr     ah,1            ;put left and right shift in lsb
        or      al,ah           ;put the lsb's together
        and     al,1            ;isolate shifted status
        mov     shftkey,al      ;pass shift bit back
        shr     ah,1            ;put control bit in lsb
        and     ah,1            ;isolate control status
        mov     ctrlkey,ah      ;pass control bit back
;next check for an actual key
        mov     ah,1            ;command to check buffer
        int     22              ;keyboard i/o
        mov     ax,0            ;command to read buffer / false return
        jz      kretr           ;no key typed, return false
        int     22              ;keyboard i/o
        mov     keycode,al      ;return key code
        mov     keyscan,ah      ;return scan code
        not     cl              ;force clear bits we forced set
        and     byte ptr es:[bx],cl   ;reset bits directly
        mov     ax,1            ;true return
kretr   label   near
        pop     bp
        ret
keytype endp

keyinit proc    far
        les     bx,flagads      ;address bios shift flags
        or      byte ptr es:[bx],030H ;setup num lock and scroll lock
        ret                     ;finished
keyinit endp

keyterm proc    far
        les     bx,flagads      ;address bios shift flags
        and     byte ptr es:[bx],0CFH ;clear num lock and scroll lock
        ret                     ;finished
keyterm endp

public  mover,movel
;various bytestream move routines;some better than standard
;
;procedure mover (source, target: adrmem;length: word);
;  move len bytes from source to target, starting at right end.
mover   proc    far
        pop     bx              ;return offset
        pop     dx              ;return segment
        mov     ax,ds           ;setup segment
        mov     es,ax           ;for target di
        pop     cx              ;byte length
        pop     di              ;target offset
        add     di,cx           ;start at end
        dec     di              ;last target byte
        dec     di              ;last target word
        pop     si              ;source offset
        add     si,cx           ;start at end
        dec     si              ;last source byte
        dec     si              ;last source word
        std                     ;direction backward
        shr     cx,1            ;make into word count
        jz      movrb           ;skip if no words to move
        rep     movsw           ;move si-- to di-- for cx
movrb:  jnc     movrf           ;finished if even count
        inc     si              ;to source byte
        inc     di              ;to target byte
        movsb                   ;move odd byte
movrf:  push    dx              ;return segment
        push    bx              ;return offset
        cld                     ;leave forward
        ret                     ;finished
mover   endp
;
;procedure movel (source, target: adrmem;length: word);
;  move len bytes from source to target, starting at left end.
movel   proc    far
        pop     bx              ;return offset
        pop     dx              ;return segment
        mov     ax,ds           ;setup segment
        mov     es,ax           ;for target di
        pop     cx              ;byte length
        pop     di              ;target offset
        pop     si              ;source offset
        cld                     ;direction forward
        shr     cx,1            ;make into word count
        jz      movlb           ;skip if no words to move
        rep     movsw           ;move si++ to di++ for cx
movlb:  jnc     movlf           ;finished if even count
        movsb                   ;move odd byte
movlf:  push    dx              ;return segment
        push    bx              ;return offset
        ret                     ;finished
movel   endp
```

□

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SERIOUS



about your IBM PC?

***READ** the Definitive Journal for the IBM Personal Computer User.* Too serious about your IBM PC to plow through 3 lbs. of ads...industry history...revealing personal interviews? Consider Personal Computer Age. Written and published monthly by serious computerists for serious computerists, Personal Computer Age presents PC technology from fundamentals to hands-on tweaking...And explains it so you can use it.

Start your subscription now. Personal Computer Age... one year 12 informative issues...just $24.00.

The Definitive Journal for the IBM Personal Computer User
Personal Computer Age

Enter my subscription today!

☐ New
☐ Renewal
☐ Payment enclosed
☐ Bill me

☐ 1 yr. (12 issues) $24.00
☐ 2 yrs. (24 issues) $45.00
☐ 3 yrs. (36 issues) $64.00
☐ Send information on back issues, too!

charge my:
☐ Visa ☐ MASTERCARD Exp Date_____

Card # _____

_____
(authorization signature)

1 yr. subscription: Canada & Mexico $29.00
Other countries via Air . $90.00
Surface mail $40.00
Payment made in U.S. funds drawn on U.S. bank.

Name _____

Address _____

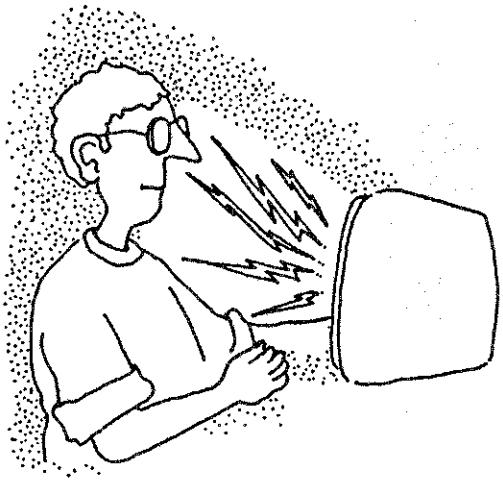
City _____ State _____ Zip _____

Department 6083
P.O. Box 70725
Pasadena,
California 91107
(800) 824-7888
operator 77

The Definitive Journal for the IBM Personal Computer User
Personal Computer Age

Department 6083
P.O. Box 70725
Pasadena,
California 91107
(800) 824-7888
operator 77

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



by Charles Wallach

VODS

VDT/VODS BULLETIN

It is widely reported that individuals who use Cathode Ray Tube (CRT) types of computer terminal displays or word processors (VDTs) are often subject to one or more of a variety of complaints classed as Video Operator's Distress Syndrome (VODS) under the following conditions:

* Frequent daily use
* Continuous use for more than two hours at a time
* Operator's face within 18 inches of display surface
* Little or no cross ventilation
* No conductive shield over the face of the CRT

It is emphasized that occasional, casual use of these devices appears to present no such problems, and that the fault is not with the VDT if used according to the manufacturer's recommendations (as VDTs are subject to strict radiation control), but rather the fault is with the environment in which the VDT is used.

VODS effects vary over broad ranges of symptoms and intensities with individual operator differences. In general, females appear to be more sensitive than males, and sensitivity tends to increase with age. Some operators appear to be completely unaffected; with others, the effects may be too subtle and slow in developing to be readily noticed, but nevertheless exist. Pregnant operators may be at particular risk because they are unable to sense the effects of VODS on the fetus or on the course of their pregnancy. After two hours of continuous operation under the conditions cited above, a large fraction of daily operators in the sensitive category may notice one or two (but never all) of the typical VODS symptoms, which include the following: headache or the feeling of an impending headache, loss of normal appetite or actual nausea, respiratory distress due to asthma or allery triggering hyper-sensitivity to cigarette smoke and other air pollutants, heightened stress (affecting fetal development), unusual irritability and moodiness, early onset of fatigue, muscular stiffness or pain in the neck or back, shortened attention span, loss of interest in work, abnormal susceptibility to colds or flu, reduced productivity, reddening, itch or rash of the facial skin and eyes.

Of course there are many other factors which can produce any one of these symptoms; however, when one or two appear consistently after an hour or two of VDT operation under the above conditions, it is probable that VODS is involved.

We believe that VODS is caused by a phenomenon known as "ion depletion" in the ambient air. Laboratory mammals kept in ion-depleted environments suffer very high mortality rates, and are unusually susceptible to diseases they can normally resist. Humans remaining in stuffy, ion-depleted rooms for extended periods become moody, depressed, and subject to exactly the same menu of symptoms (except the last one) described by sensitive VDT operators.

The one thing that all CRT-type VDTs have in common is a high positive voltage, or electrostatic charge on the face of the screen. This attracts the negative ions out of the near airspace, and at the same time repels all the positive ions beyond the operator's breathing range, thus eliminating the natural ion balance and population from the ambient air — and these small ions are a biologically important component of fresh air, which have been too long taken for granted and neglected in environmental research.

The same positive charge also repels like-charged particles of various air pollutants toward the face of the operator with considerable velocity, leading to irritation of sensitive skin and membranes from this "micro-bombardment" process. Theoretically, these problems exist with ordinary TV sets and video games, but usually the viewers are far enough away from the screen that the normal currents of room air carry enough ions into their respiratory airspace. Some particularly sensitive individuals may feel uncomfortable when they watch TV for an extended length of time; but the serious symptoms of VODS are chiefly found to be associated with VDTs and radar displays.

Most VODS problems can be eliminated entirely by re-ionizing the air breathed by the operator, or by eliminating the electrostatic charge on the face of the display tube, or both. There are around 60 negative ion generator (NIG) models on the U.S. market, of which only about a dozen are worth considering. Normally, ion depletion is best remedied by putting a good NIG where the negative ions will drift or be blown between the VDT and the operator, or better yet, using a model which

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

combines the NIG with a fan that directs the ion stream.

However, and this is very important, where there are microprocessors or computer elements in the VDT adjacent to it, a NIG may generate enough local static to interfere with the electronics. In this case, a Bipolar Ion Generator (BIG) must be used to provide both positive and negative ions in sufficient quantities to keep the environment entirely static-free. At this writing, we know of only two established and reputable sources of BIGs. Ion Systems Inc. offers a ceiling-suspended system to be installed directly over the operator's position, which requires special wiring and installation by an electrician. Biotech Electronics Ltd. has developed a BIG in a breadbox-size unit which includes a low-noise, variable-speed blower and a highly effective filter to remove pollutants and odors — this can be placed on a nearby table or shelf, and plugged into a normal socket. See enclosed material for addresses and a selection of NIGs; BIG models are quoted directly in a configuration most suited to your requirements, by the manufacturers or their agents.

The ion depletion problem can also be corrected by having a grounded, conductive screen installed over the face of the CRT. The VDT must be disassembled for this, and requires a knowledgeable technician to do it properly. These screens are not sold by mail or in stores, but through trained representatives of Sun-Flex Inc., 20 Pimentel Ct., Novato, CA 94947, (415) 883-1221. The Sun-Flex screen also makes it possible to relieve eyestrain by improving the contrast of the characters on the VDT. One or two VDT manufacturers are beginning to install these special, patented screens as standard items, and it is hoped that eventually all of them will do this.

By combining the Sun-Flex screen and a BIG, you may enjoy the best of both worlds, as both products have singular healthful advantages apart from completely eliminating the environmental cause of VODS. Both technologies have long been used by clear-sighted problem solvers with success and satisfaction, although their effectiveness has not yet been recognized by U.S. authorities despite wide acceptance in Europe. In fact, it may be largely at our instigation that such organizations as the National Institute for Occupational Safety and Health (NIOSH) and related agencies have moved from their previous position of rejecting the concept of ion depletion effects on human biochemistry, to their current official position of indicating that there is not yet sufficient proof for regulatory action — which will require years of costly testing to satisfy the stringent proof requirements of U.S. agencies. Because of other, higher priorities, they have not yet been funded to pursue the ion-depletion aspects of the serious and growing VODS problems, especially as these may effect pregnancies, and are understandably reluctant to contribute to irrational fears of VDT use. Also, NIOSH has apparently confused environmental ion effects with VDT radiation effects which are completely unrelated, and insignificant. Perhaps the public pressure of concerned users will cause VODS to be seriously addressed at higher priorities as has been done in other countries.

NOTE: THE SEVERITY OF VODS MAY BE REDUCED BY USING A CONSUMER-TYPE NEGATIVE ION GENERATOR AT HOME, ACCORDING TO SOME OPERATORS.

The above excerpt is from a bulletin from IBEF. For more information on VODS and ion depletion send a self-addressed, stamped business envelope to:

Charles Wallach, Exec. Director
International Bio-Environmental Foundation
Box 2533
Northridge, CA 91323
(213-882-7128)

Mr. Wallach included a letter which noted:

As far as the research is concerned, we have a number of reports from all over the world. These are listed in the "Ion Bibliography" available for $20 ppd from Decisions & Designs, Box 907, McLean, VA 22101 (120pp annotated).

For your information, the most successful ionization technique is to use a generator within four or five feet of the operator, off to one side so that it blows its ionized, filtered zephyr between the operator's face and the tube. The constant flow of negative ions drawn toward the tube by its positive charge tends to keep that charge near zero with plenty of ions left over for the operator to breathe. At work stations where this is not convenient, a larger unit further away may be needed.

In the same mail as your letter comes a report from a lady operator, 32, who had the pattern of working for three days in front of a Xerox 860, and coming down with an asthma attack on the fourth day. She had to take medicine for the asthma, and the side effects made her feel so miserable she'd stay home for a day, and then resume the cycle. Her operation was a 7-day/week one. She got a Bionaire 500 ionizer, and from the day she started using it (about a month ago) she hasn't had a single asthma attack on the job. A few days ago, on her normal day off, there was a medium-sized Santa Ana wind blowing in her area (loaded with positive ions). She started to go shopping, but came down with an asthma attack. So she rushed to work, on her day off, because that's where her ionizer was, and she felt fine almost immediately. Wow! We're going to frame her letter in the front hall. □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

COMPILED BASIC — THE NEXT STEP

by Ronald Moak

Despite the cries of many in the field of computing, BASIC is still the defacto language of the microcomputer. The fact that BASIC is the only language available on most home and small business computers virtually insures its survival through the next few years. For the individual who is considering learning to program, there is a definite lack of good introductory books; however, one need go no farther than the local bookstore to become overwhelmed by the number of BASIC programming texts. They seem to be springing forth daily on every subject from graphics to engineering.

For the majority of programmers, BASIC will be the language of choice. It is a relatively easy language to learn, compared to other languages like C or Pascal. The IBM PC's interpreter with on-screen editing makes BASIC especially attractive to the trial-and-error style of beginning programmers.

Unfortunately, the advantages of BASIC quickly disappear with increased program complexity. Structured programming conventions involving IF-THEN-ELSE statements are usually simple at first, but as the program grows, so does the complexity of the code. The following example is typical of the complexity that a BASIC program can have.

```
705 IF ASC(INK$)=9 THEN IF C=6 THEN V=1:GOSUB 800:C=29:RETURN
    ELSE IF C=29 THEN V=2:GOSUB 800:C=35:RETURN
    ELSE IF C=35 THEN V=3:GOSUB 800:C=37:RETURN
    ELSE IF C=37 THEN IF R=FIRST THEN V=4:GOSUB 800
    :R=4:C=6:RETURN ELSE V=4:GOSUB 800:R=R+1:RETURN
```

One solution to unreadable code is to break it up. Write only one statement per line and use GOTOs to branch around unnecessary code. On the one hand this may produce more readable code, but it also slows the program down. The real problem lies in the fact that BASIC limits lines to 254 characters and requires continuous lines. This forces the programmer to use short variable names (long variable names take up more line space) and write code that can be unreadable two minutes after it is written.

Another solution to the complex programs and unreadable code situation is to pick a new programming language. But what happens to the time and effort that went into learning BASIC? If your job is software development, then you can justify learning a new language. Most people feel they have invested too much time studying BASIC in the first place.

Numerous articles and books have been published over the last couple of years on structured BASIC, so I will not discuss it here. Instead, this article will cover ways to improve your code by using a specific tool, the IBM BASIC compiler.

BASIC COMPILER OPTIMIZING

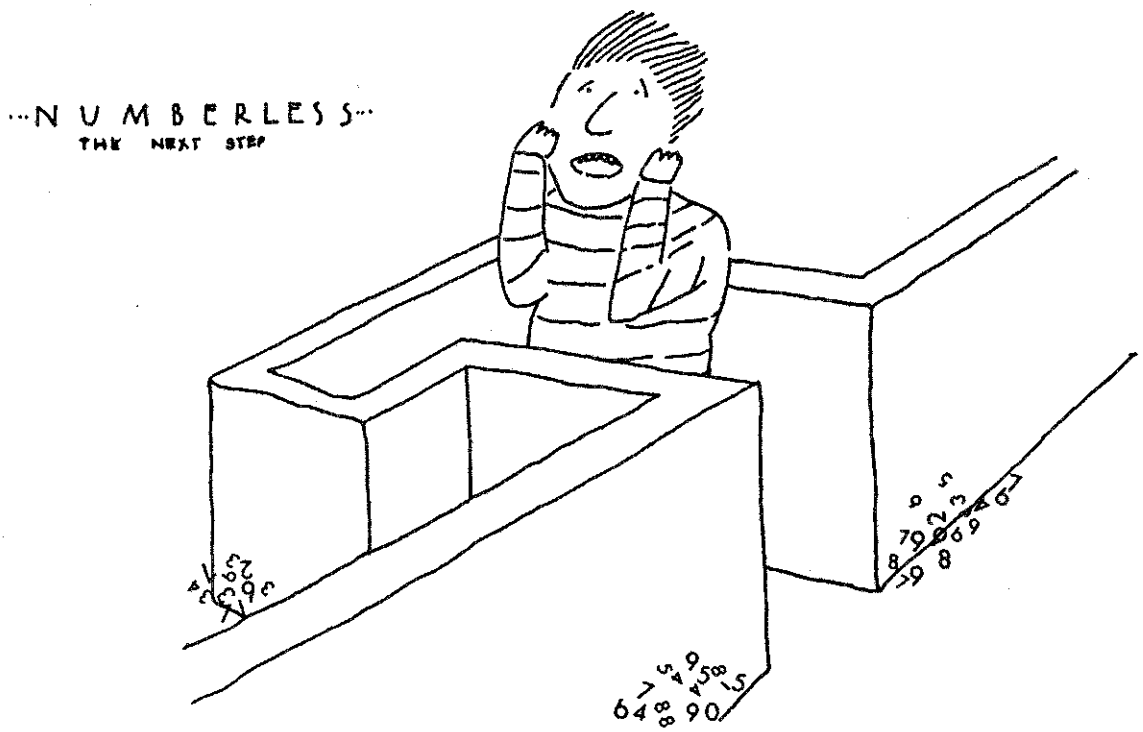
Many people view the compiler only as a means to speed up the execution of existing BASIC programs. While this is certainly true, the compiler can help in the structuring of your code as well. It limits lines to 32,767 characters instead of 255 like the interpreter. This allows the IF-THEN-ELSE structure, normally limited to one line, to be very long with multiple branches. Because most line editors do not allow lines of 32,767 characters, the compiler reserves the underline character (_) to indicate the continuation of a logical line. The IF-THEN-ELSE structure illustrated above was rewritten into a form still acceptable to the compiler but much more readable to the programmer.

Listing 1:

```
705 IF ASC(INK$) = 9 THEN_                    'test input
        IF C = 6  THEN_                       'if column = 6
            V = 1:_
            GOSUB 800:_                       'read values
            C = 29:_
            RETURN_
        ELSE_
            IF C = 29 THEN_                   'if column = 29
                V = 2:_
                GOSUB 800:_                   'read values
                C = 35:_
                RETURN_
            ELSE_
                IF C = 35 THEN_               'if column = 35
                    V = 3:_
                    GOSUB 800:_               'read values
                    C = 37:_
                    RETURN_
                ELSE_
                    IF C = 37 THEN_           'if column = 37
                        IF R = FIRST THEN_    'if row = one
                            V = 4:_
                            GOSUB 800:_       'read values
                            R = 4:_
                            C = 6:_
                            RETURN_
                        ELSE_
                            V = 4:_
                            GOSUB 800:_       'read values
                            R = R + 1:_
                            RETURN
```

Listing 1 illustrates an important feature of the BASIC compiler. The continuation character tells the compiler to look to the next line for additional statements. This also allows you to comment each line. Comments placed after the continuation character are ignored by the compiler. Under the interpreter, the only way to comment a single line is to place the comment at the end of the line. During most day-to-day programming, I would not break up the four-line BASIC code into 31

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



lines; this code was expanded to illustrate the extent to which you can separate unreadable code into a more structured and legible format. The ability to place comments on a logical line is an invaluable tool in developing programs that are easy to debug.

The colon is required between statements to separate them on the logical line. If a colon is left out between statements at the end of a line with the continuation character inserted, then the compiler will flag it as an error. If both are missing, the compiler assumes the end of the logical line. As with the interpreter, colons are not required before an ELSE statement.

A combination of long logical lines and structured statements allows the careful programmer to write in BASIC without using GOTO statements which result in spaghetti code. (Listing 4 is the same as Listing 3 except for the GOTO structures, which were changed to a WHILE-WEND loop.)

Another attractive feature of the compiler is the ability to write code without line numbers. Since the compiler translates the code only once at the time of compilation, it does not require line numbers, except when branching to a particular statement or subroutine. Interpretive BASIC, on the other hand, does require line numbers.

Subroutines may be written and referenced with a single number at the beginning of the routine; this eliminates the possibility of branching into the middle. Programs coded without line numbers are easier to read because indented structures stand out. Blank lines may be placed in the code to separate different functions.

A second advantage to writing code without line numbers occurs during translation when the compiler optimizes the execution between line numbers. In other words, all statements between numbers are organized to determine the fastest method of execution.

Table 1 is a listing of running times for the Sieve program. The lack of line numbers increased execution speed by .1 of a second. Table 1 also compares the execution speed of BASIC programs compiled with different options. The Eratosthenes Sieve Prime Number Program was used to compare runtime speeds. The Sieve program is often criticized when used to test the speed of different computers. In this case the program is simply used to show the relationships between the various compiler options.

The greatest difference in program execution occurred when the program was compiled with event trapping. The version without line numbers executed twice as fast when nonessential line numbers were removed. The primary reason for the large difference is the fact that under the /W option the program tests for event trapping only where line numbers occur. If you want event trapping to take place at every statement, then the /W is replaced with a /V. Structured programs without line numbers take about the same time to run as programs with line numbers and one statement per line (both compiled with event trapping on each statement.)

When you invoke event trapping just where line numbers occur, this will be the only time that the program checks for function key action. If you trap at each statement, the test must be performed more often. One of the side effects of this last method is that four extra bytes of code are attached to each statement, rather than to each line number. If you also utilize the RESUME-NEXT option on a return from error trapping, then

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ANOTHER four bytes of code are included for a total of eight bytes per statement. On a large program, the extra four or eight bytes may determine whether it will successfully compile within BASIC's 64K code segment restriction.

Table 1 illustrates the speed differences between compiled programs that use the runtime library and programs designed as stand alone modules. Programs that do not use the runtime library are compiled with the /O option.

Table 1
Benchmark test of Compiled BASIC
Times listed in seconds

Option	Listing 2	Listing 3	Listing 4
N/O	1.7	1.6	1.6
N/O/E	1.7	1.6	1.6
N/O/E/W	5.4	2.6	1.7
N	2.2	2.1	2.1
N/E	2.2	2.1	2.1
N/E/W	6.0	3.1	2.1

Time for Benchmark run under interpretive BASIC is 4 min. 20 sec.

Option descriptions:

N - Line numbers used only as labels for branching.
O - Stand-alone version does not require runtime module when executing.
E - Enables error trapping within program.
W - Enables event trapping within program.

Program listings 2 and 3 are identical except that program 3 has all unnecessary line numbers removed.

Listing 2: Standard BASIC Interpretive

```
1'  Eratosthenes Sieve Prime Number Program
5'
10   defint a-z
20   size = 8190
30   dim flags(8191)
40   print "Ten Interpretive Iterations "
50   def fnsec(SEC) = val(right$(time$,sec))
60   begin_time = fnsec(2)
65   for m = 1 to 10
70       count = 0
80       for i = 0 to size
90          flags(i) = 1
100      next
110      for i = 0 to size
120          if flags(i) = 0 then 200
130          prime = i + i + 3
140          k = i + prime
150          if k > size then 190
160          flags(k) = 0
170          k = k + prime
180          goto 150
190          count = count + 1
200      next
205  next
210  end_time = fnsec(2)
220  print "number of primes = "; count
230  if end_time <= begin_time then end_time = end_time + 60
240  print "Average seconds per iteration = ";
260  print (end_time - begin_time) / 10
250  end
```

Listing 3: Prime Number Program using line numbers only for labels

```
'  Eratosthenes Sieve Prime Number Program
'
     defint a-z
     size = 8190
     dim flags(8191)

     print "Ten Line Numberless Iterations "
     def fnsec(SEC) = val(right$(time$,sec))
     begin_time = fnsec(2)

     for m = 1 to 10
        count = 0
        for i = 0 to size
            flags(i) = 1
        next
        for i = 0 to size
           if flags(i) = 0 then 200
           prime = i + i + 3
           k = i + prime
150        if k > size then 190
                flags(k) = 0
                k = k + prime
                goto 150
190        count = count + 1
200     next
    next

    end_time = fnsec(2)
    print "number of primes = ";count
    if end_time < begin_time then end_time = end_time + 60
    print "Average seconds per iteration = ";_
     (end_time - begin_time) / 10;
    end
```

If your work has been primarily in BASIC, then learning to write structured code without line numbers will ease your transition to other numberless languages. Writing code without BASIC interpreter constraints will greatly increase your program's readability. The reduction in hours needed to debug spaghetti code should more than offset the extra time required with a compiler and editor. The payoff really shows up when you need to work on a program you have not seen for six months.

Listing 4: Structured BASIC Program with no line numbers

```
'  Eratosthenes Sieve Prime Number Program
'
defint a-z
size = 8190
dim flags(8191)
print "Ten Structured Iterations "
def fnsec(SEC) = val(right$(time$,sec))
begin_time = fnsec(2)
for m = 1 to 10
    count = 0
    for i = 0 to size
        flags(i) = 1
    next
    for i = 0 to size
        if flags(i) then          _
            prime = i + i + 3     :_
            k = i + prime         :_
            while (k <= size)     :_
                flags(k) = 0      :_
                k = k + prime     :_
            wend                  :_
            count = count + 1
    next
next
end_time = fnsec(2)
print "number of primes = "; count
if end_time < begin_time then_
     end_time = end_time + 60
print "Average seconds per iteration = ";_
 (end_time - begin_time) / 10
end
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ASSEMBLY MODULES FOR C

by Ted Reuss

Ted has developed a number of applications using the C language (Lattice/Microsoft). In the process, he designed a set of assembly routines to provide the library services he needs, besides altering code to allow PATH names and chaining among C programs. This issue includes a number of Ted's assembly routines, and a following issue will take a look at the C functions which call them.

```
title   Video Routines for Lattice 'c'  10.2.83
page    64,132
name    VIDEO
pgroup  group   prog
prog    segment byte public 'prog'
        public  VATTR,VCLS,VCURG,VCURP,VCURS,VPUT,VSCROL
        assume  cs:pgroup

MONO_SEG  = 0b000h              ;Mono card segment
COLOR_SEG = 0b800h              ;Color card segment
;
; name      vattr -- sets character attributes
;
; synopsis  void = vattr(attr,count,row,col);
;           char   attr;
;           short  count;
;           short  row;     row (0-24)
;           short  col;     column (0-79)
;
; description   This function sets the attribute for count characters
;               starting at row,col. Cursor position is not changed.
;
VATTR   proc    near
        push    bp
        mov     bp,sp
        mov     ah,0fh
        push    bp
        int     10h             ;BIOS video call
        pop     bp
        cmp     al,7            ; mono card?
        mov     bx,MONO_SEG
        je      vat10           ; YES
        mov     bx,COLOR_SEG    ; NO
vat10:  mov     dx,es           ;save ES
        mov     es,bx
        mov     ax,[bp+8]       ;Get row to AX
        mov     ah,160          ;row * 160
        mul     ah
        mov     di,[bp+10]      ;Get col to DI
        shl     di,1            ;col * 2
        add     di,ax
        inc     di
        mov     ax,[bp+4]       ;AX=attr
        mov     cx,[bp+6]       ;CX=count
        cld
vat20:  stosb
        inc     di
        loop    vat20
        mov     es,dx           ;restore ES
        mov     sp,bp
        pop     bp
        ret
VATTR   endp

;
; vcls() -- clear screen & position cursor to 0,0
;
VCLS    proc    near
        push    bp
        mov     bp,sp
        xor     ax,ax
        push    ax
        push    ax
        call    vcurp
        pop     ax
        pop     ax
        mov     ax,0600h
        mov     cx,0
        mov     dx,7924h
        mov     bh,7
        push    bp
        int     10h
        pop     bp
        mov     sp,bp
        pop     bp
        ret
VCLS    endp
;
; name      vcurg -- get cursor position & type
;
; synopsis  void = vcurg(row,col,beg,end);
;           short  *row;    row (0-24)
;           short  *col;    column (0-79)
;           short  *beg;    begin (0-13)
;           short  *end;    end (0-13)
;
; description   This function returns the cursor position
;               & type as a packed long integer.
;
VCURG   proc    near
        push    bp
        mov     bp,sp
        xor     bx,bx
        mov     ah,3            ;read cursor position & type
        push    bp
        int     10h             ;BIOS video call
        pop     bp
        xor     ax,ax
        mov     bx,[bp+4]       ;get &row
        mov     al,dh
        mov     [bx],ax         ;store it
        mov     bx,[bp+6]       ;get &col
        mov     al,dl
        mov     [bx],ax         ;store it
        mov     bx,[bp+8]       ;get &beg
        mov     al,ch
        mov     [bx],ax         ;store it
        mov     bx,[bp+10]      ;get &end
        mov     al,cl
        mov     [bx],ax         ;store it
        mov     sp,bp
        pop     bp
        ret
VCURG   endp
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
;
; name          vscrol -- scroll the specified window
;
; synopsis      void = vscrol(nl,rt,ct,rb,cb,attr)
;               short nl;       number of lines to scroll
;                               } 0 scroll up
;                               ( 0 scroll down
;                               = 0 scroll entire window
;               short rt,ct;    row & column of top left
;               short rb,cb;    row & column of bottom right
;               short attr;     attribute of blanked lines
;
; description   This function scrolls the specified window
;               either up or down by the number of lines
;               specified.  The blanked lines or given the
;               specified attribute.
;
VSCROL  proc    near
        push    bp
        mov     bp,sp
        mov     ax,[bp]+4       ;get nlines
        or      ax,ax
        mov     ah,6            ;scroll up function
        jns     vsc10           ;if nlines .ge. 0
        inc     ah              ;scroll down function
        neg     al
vsc10:  mov     cx,[bp]+6       ;get rtop
        mov     ch,cl
        mov     bx,[bp]+8       ;get ctop
        mov     cl,bl
        mov     dx,[bp]+10      ;get rbot
        mov     dh,dl
        mov     bx,[bp]+12      ;get cbot
        mov     dl,bl
        mov     bx,[bp]+14      ;get attr
        mov     bh,bl
        push    bp
        int     10h             ;BIOS video call
        pop     bp
        mov     sp,bp
        pop     bp
        ret
VSCROL  endp

prog    ends
        end


;
; name          vcurp  -- position cursor
;
; synopsis      void = vcurp(row,col);
;               short row;      0-23
;               short col;      0-79
;
; description   This function positions the cursor at the
;               specified row and column.
;
VCURP   proc    near
        push    bp
        mov     bp,sp
        mov     ax,[bp]+4       ;get row
        mov     dx,[bp]+6       ;get col
        mov     dh,al
        xor     bx,bx           ;set page number = 0
        mov     ah,2            ;set cursor position function
        push    bp
        int     10h             ;BIOS video call
        pop     bp
        mov     sp,bp
        pop     bp
        ret
VCURP   endp
;
; name          vcurs -- set cursor type
;
; synopsis      void = vcurs(beg,end);
;               short beg;      beginning scan line (0-13)
;               short end;      ending scan line (0-13)
;
; description   This function sets the cursor type.
;
VCURS   proc    near
        push    bp
        mov     bp,sp
        mov     cx,[bp]+6       ;get end
        mov     bx,[bp]+4       ;get beg
        mov     ch,bl
        mov     ah,1            ;set cursor type function
        push    bp
        int     10h             ;BIOS video call
        pop     bp
        mov     sp,bp
        pop     bp
        ret
VCURS   endp


;
; vput (char,count,attr) -- write count char's with attr
;       Defaults to 1 character.
;       If attr ( 0 then it is or'ed with the
;         current attribute.
;
VPUT    proc    near
        push    bp
        sub     bp,sp
        mov     dx,bp           ;DX = #args*2 + 4
        mov     bp,sp
        mov     ax,[bp]+4       ;get char
        mov     cx,1            ;set default count
        mov     ah,0ah          ;write char only function
        cmp     dx,6
        je      vpu10           ;if 1 arg
        mov     cx,[bp]+6       ;get count
        cmp     dx,8
        je      vpu10           ;if 2 args
        dec     ah              ;write char/attr function
        mov     bx,[bp]+8       ;get attr
        or      bx,bx
        jns     vpu10           ;if positive
        neg     bx              ; make it positive
        or      al,80h          ;set to "OR" attr
vpu10:  push    bp
        int     10h             ;BIOS video call
        pop     bp
        mov     sp,bp
        pop     bp
        ret
VPUT    endp
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DATABASE RELATIONS - SELECT, LOCATE, INDEX and FIND

by Jim Bohle

If you have ever implemented a database application in dBase II, you probably appreciate the ease and speed with which you can create files and query report information. In most cases, it's a matter of minutes. Furthermore, when you want to automate the process, your set of commands can conveniently be organized into a program using dBase II's powerful, structured database language. Even so, implementing of small database applications involves more than simply sitting down and issuing CREATE, APPEND, EDIT, REPLACE and REPORT commands.

dBASEII is promoted as a relational database system. What this means and how it is constructed is the key to designing a flexible database application. A rigorous definition of relational database organization is beyond the scope of this article; if you desire more depth try C.J. Date's book, "An Introduction to Database Systems."

Put in simple terms, the relational database approach to applications involves defining simple, consistently structured files, along with a set of relationships defining the interactions between these files. The advantages of a good relational design are many:

* Simplicity in maintaining and accessing files since, the data structure used involves simple tables.

* Flexibility in changing file structures and content.

* Ease of adding new files and relations, because of the independent nature of the information.

* Improvement in the consistency and integrity of information.

* Elimination of data redundancy by suggesting single information sources and by providing functional relations between files.

* Flexibility and utility in extracting information, since sources are logically related without reference to the details of physical storage and access (e.g. previous and next record pointers).

dBase II provides the means to define and create simple file structures (essentially tables) and the relationships which logically connect these files, although some may argue that the limitation of two active databases is a distinct disadvantage in the development of more complex applications. Relations are verified by querying associated files for the existence of specific key information (e.g. a company name or employee number). Several dBase II commands provide this capability, namely: SELECT, LOCATE, FIND, UPDATE, JOIN. The central theme of this article is the demonstration of a relational database design and how some of the supporting dBase II commands are used.

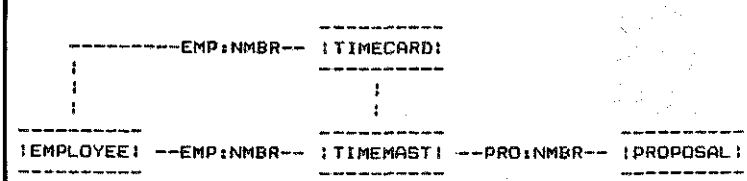
A few weeks ago, a friend mentioned a small database application he decided to implement, to teach himself dBASE II. While reasonably simple, the application provides a good framework for demonstrating a relational file organization and the use of the SELECT, LOCATE, INDEX and FIND commands (we will discuss the UPDATE and JOIN commands in another article). Specifically, the application involves the need to track on-going costs associated with a variety of project proposals undertaken by the engineering firm for which he works. The central information source is a weekly batch of employee time cards containing an employee number, a proposal number, and the hours worked for the three phases of a proposal. A cumulative time card database is maintained, the primary output being a summary of hours and costs for each employee involved in the proposal.

We discussed the application and several possible approaches. The initial attempt used a single file containing all the required information. It became obvious this approach lacked the simplicity and flexibility of employing a set of simple related files and introduced a great deal of data redundancy, another potential for erroneous information. It also seemed that changing the application or adding enhancements would require a fair amount of labor. Eventually, we noticed the independent nature of several pieces of information. The employee information could be maintained independently from both the proposal and timecard information. Using the capabilities of dBase II we could easily relate these independent files to provide the requested information. Furthermore, each database could be used independently from the others, yet serve as the single central information source for other databases (e.g. validation of employee number when entering timecard information or retrieving the employee name for report purposes). Ultimately, four databases were defined for the application:

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
EMPLOYEE database
-----------------
EMP:NMBR N 4 employee id, unique key, index ENMBR.NDX
EMP:NAME C 25 employee name; may want first, last separate
HRLY:RATE N 5,2 employee hourly pay rate (cost rate)



PROPOSAL database
-----------------
PRO:NMBR C 7 proposal id, unique key, index PNMBR.NDX
PRO:NAME C 25 proposal name or description
PRO:CLIENT C 25 proposal client name or other reference
PRO:FEE N 10,2 projected fee for proposal
PRO:ODC N 8,2 other direct costs associated with proposal
PRO:STATUS C 1 proposal status; e.g. active or complete



TIMECARD and TIMEMAST databases
-------------------------------
EMP:NMBR N 4 employee id
PRO:NMBR C 7 proposal id; part of index TCPDATE.NDX
PROP:HRS N 4,1 hours worked on proposal phase
PRES:HRS N 4,1 hours worked on presentation phase
NEGO:HRS N 4,1 hours worked on negotiation phase
ENTRY:DATE C 8 entry/batch date; part of index TCPDATE.NDX
```

This database organization demonstrates simplicity in structure, file independence, and absence of data redundancy, yet provides the flexibility (through relations) to produce the desired query and report capabilities. The following database relations are defined:



Four dBase II programs were written to perform the desired database maintenance and reporting functions (please refer to the program listings below). Our purpose in presenting these programs is to demonstrate the facilities available in dBase II to implement and verify relations between independently-organized files, and to illustrates some of the advantages associated with viewing database applications from a relational standpoint.

PROGRAMS

The programs EMPLOYEE.PRG and PROPOSAL.PRG are essentially identical in function in that they provide maintenance of the EMPLOYEE and PROPOSAL databases, respectively. Depending on the particular operation selected, the program must search the database on the key field (either EMP:NMBR or PRO:NMBR) to check for duplicate entries (for Add) or locate the requested record (for Change and Delete). Three techniques for searching the database come to mind. The first is what could be called a 'manual' search since the programmer must write the commands for loop control and key value comparison. A manual search routine to check for duplicate employees in EMPLOYEE.PRG could be used in place of the marked (—>) statements and might look like the following:

```
store F to FOUND
goto top
do while .not. eof .and. (.not. FOUND)
   if E:NUM=EMP:NMBR
      store T to FOUND
   else
      skip
   endif
enddo
if .not. FOUND
```

The manual search is considerably slower (in part due to the increased number of command statements and the interpretive nature of dBase II) and certainly more work than either of the other two methods. The LOCATE command provides a non-procedural approach to performing the manual search. The command has the following format:

locate [<scope>] [for <expression>]

LOCATE is generally used with the default 'all' scope and always begins searching with the first record in the database. The expression denotes the specific relation or condition on which the search is based. The manual search could be replaced with the LOCATE command in EMPLOYEE.PRG as follows:

```
locate for E:NUM = EMP:NMBR
if E:NUM<>EMP:NMBR ...
```

The LOCATE command is simple to use and reasonably fast. It does not require index files and in fact works much quicker with non-indexed databases since it performs a sequential search. The expression provides a good deal of flexibility for specifying a variety of relationships (other than equality) which is not available with the FIND command. The major disadvantages with LOCATE are its need to start searching at the beginning

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

of the database (unless a CONTINUE can be used,) and, as the database grows in size, the LOCATE command becomes noticeably slower.

The search strategy actually used in each of the four programs is based on the FIND command which has the simple format:

find <object>

The FIND command uses objects which are either numeric constants (e.g. 12, 577), character strings (e.g. SMITH, '2A6') or character string memory variables (e.g. &KEY, &EMPNUM). The latter is the typical form used in programs where variable key information is used. Please note: the memory variable must be a character type so numeric data items must be converted to strings using the 'str' function. The '&' is a macro operator and simply means 'use the contents of the memory variable'. Therefore, programs generally store the key value (e.g. EMP:NMBR or PRO:NMBR) to a memory variable which is used in the FIND command. FIND has a number of distinct advantages over the other two search methods: first, it is very fast, virturally instantaneous, due to the use of indexes based on b-tree structures and is largely unaffected by database size, and second, it is more efficient since it reads an index file to find the desired key rather than reading database records (which are undoubtedly larger). The FIND command does require an index (ENMBR.NDX and PNMBR.NDX in this example). Once described, however, both programs can automatically maintain the indexes (as records are added, changed or deleted) if the database is used with the index set. The commands to create the ENMBR and PNMBR index files are:

index on EMP:NMBR to ENMBR.NDX
index on PRO:NMBR to PNMBR.NDX

The TIMECARD program illustrates the use of the FIND command to verify the relation between the TIMECARD and EMPLOYEE databases. As employee timecard information is entered, each employee id (EMP:NMBR) is validated by FINDing it in the EMPLOYEE database. The SELECT command is used to indicate which of the two databases is currently being referenced during program execution. If the entered employee ID is found, the employee's name and hourly rate are displayed on the screen to further aid validation. If the employee ID cannot be found, the record position function, '#', yields a zero and the program duly informs the user. The LOCATE command could be used in lieu of the FIND in the same way as shown in EMPLOYEE.PRG, but if the EMPLOYEE database has over 20-25 records, a noticeable delay occurs.

Some comments on the TIMECARD program. First, validation of proposal numbers is not performed, since timecard information may be entered prior to creation of a corresponding record in the PROPOSAL database. However, if validation was thought to be necessary, the process to find the specific proposal is nearly identical to the one for employee IDs. The only hitch is the need to alternate the secondary database between the EMPLOYEE and PROPOSAL databases. This is an example of where dBase II introduces operational inefficiency by permitting only two active databases at any one time. The use of three databases is demonstrated in the next program, PREPORT.PRG.

Second, it is assumed that employee hourly rates will not change, since the rate is not stored in each timecard record. If rates are expected to change, perhaps the hourly rate should be stored in the TIMECARD database. The rate used for cost calculations would be retrieved from the TIMECARD database rather than the EMPLOYEE database. If the field were named HRLY:RATE in the TIMECARD database, the following statement would update this field from the EMPLOYEE database once the correct employee record was found:

replace P.HRLY:RATE with S.HRLY:RATE

'P.' and 'S.' refer to Primary and Secondary databases, respectively. Third, as mentioned in the program listing, the batch of timecards, once entered and verified, is appended to the cumulative master timecard database. Several indexes associated with the TIMEMAST database for query and report purposes can be maintained at the time of the copy. One index could be based on the employee id (EMP:NMBR) and another on the proposal number (PRO:NMBR) coupled with the timecard entry date (ENTRY:DATE) in year-month-day order. This latter index is actually used in the PREPORT.PRG to provide the proposal cost status report and is created by the command:

```
index on PRO:NMBR + $(ENTRY:DATE,7,2) + $(ENTRY:DATE,1,2)
                  + $(ENTRY:DATE,4,2) to TCPDATE.NDX
```

The PREPORT program demonstrates the use of three databases using the FIND command to verify relations and to retrieve the report information. TIMEMAST is used as the primary database since all timecard entries should appear on the report. The EMPLOYEE and PROPOSAL databases are used to supply information for the employee ID and proposal number found in each TIMEMAST record. They are alternately selected as secondary files for each TIMECARD record.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
************     EMPLOYEE.PRG  --  Employee File Entry and Edit     ************
*                                                                              *
*        The EMPLOYEE.PRG routine provides a means for maintaining the         *
*        EMPLOYEE file.  Only the EMPLOYEE file is used.                       *
*                                 EMPLOYEE.DBF                                 *
*                            ----------------------                            *
*                            01   EMP:NMBR  N  04                              *
*                            02   EMP:NAME  C  25                              *
*                            03   HRLY:RATE N  05,02                           *
*                                                                              *
*        The first options belong to the main menu.    In the 'add' mode       *
*        a check for duplicate employee records is made prior to adding        *
*        the record to the file.  To do this a search of EMP:NMBR's must       *
*        be made.  This can be handled in several ways: manually reading       *
*        though the file using SKIP, sequentially searching by use of the      *
*        LOCATE command, or by checking an index (ENMBR.NDX) of EMP:NMBR's     *
*        with the FIND command (which is the method used here).                *
*                                                                              *
********************************************************************************

*      perform initial housekeeping and set options
erase
set talk off
set carry off
set colon off
set bell off

*      initialize memory variables
store ' ' to CHOICE
store ' ' to ANY:KEY
store '    ' to EMPNUM
store 0 to E:NUM

*      select database and set index
use EMPLOYEE index ENMBR

*      main routine loop - continue until exit is requested
do while CHOICE () 'E'
    erase
    @  4,26 say 'Employee File Maintenance'
    @  5,26 say '--------------------------'
    @  7,26 say '  A - add new employee'
    @  8,26 say '  C - change employee'
    @  9,26 say '  D - delete employee'
    @ 10,26 say '  L - list employees'
    @ 11,26 say '  E - exit'
    @ 13,26 say '  Choice ';
            get CHOICE picture '!'
    read

    do case
       routine to add a new record to the database
       case CHOICE = 'A'
            erase
            @  6,14 say 'Please enter the employee information as requested.'
            @  8,14 say 'Press (return) for Employee Number when finished.'
            @ 23,50 say 'Press any key to continue ';
                    get ANY:KEY picture "X"
            read
            store T to MORE

*      continue as long as an employee number is entered
            do while MORE
               erase
               store 0 to E:NUM
               @  1,0  say 'EMPLOYEE FILE -- ADD'
               @  2,0  say '---------------------'
               @ 10,10 say 'Employee Number ';
                       get E:NUM picture "####"
               read
               if E:NUM<>0
*      if an employee number is entered, convert to a string value
*      and try to find if it already exists
                  store str(E:NUM,4) to EMPNUM
                  find &EMPNUM
                  if #=0
*      employee number doesn't exist, so collect information and append record
                     append blank
                     replace EMP:NMBR with E:NUM
                     @ 12,10 say 'Employee Name   ';
                             get EMP:NAME  picture "XXXXXXXXXXXXXXXXXXXXXXXX"
                     @ 14,10 say 'Hourly Rate     ';
                             get HRLY:RATE picture "##.##"
                     read

                  else
*      employee number already exists, so give message and information
                     @ 10,40 say 'Duplicate Employee Number!'
                     @ 11,40 say EMP:NMBR using "####"
                     @ 11,46 say EMP:NAME
                     @ 12,46 say HRLY:RATE  using "$99.99"
                     @ 23,50 say 'Press any key to continue ';
                             get ANY:KEY picture "X"
                     read
                  endif
               else
                  store F to MORE
               endif
            enddo

       case CHOICE='C'
*        perform change employee routine

       case CHOICE='D'
*        perform delete employee routine

       case CHOICE='L'
*        perform list (display or print) employees routine

       otherwise
*        either 'E' for exit or invalid repsonse

    endcase
enddo

*      perform final housekeeping before exit
use
release all
return
```

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
************     PROPOSAL.PRG  --  Proposal File Entry and Edit     ************
*                                                                              *
*       The PROPOSAL.PRG routine provides a means for maintaining the          *
*       PROPOSAL file.  Only the PROPOSAL file is used.                        *
*                                                                              *
*                                PROPOSAL.DBF                                  *
*                              ------------------                              *
*                              01   PRO:NMBR    C  07                          *
*                              02   PRO:NAME    C  25                          *
*                              03   PRO:CLIENT  C  25                          *
*                              04   PRO:FEE     N  10,2                        *
*                              05   PRO:ODC     N   8,2                        *
*                              06   PRO:STATUS  C  01                          *
*                                                                              *
*       The first options are part of the main menu.  In the 'add' mode        *
*       a check for duplicate proposal records is made prior to adding         *
*       the record to the file.  To do this a search of PRO:NMBR's must        *
*       be made.  This can be handled in several ways; manually reading        *
*       though the file using SKIP, sequentially searching by use of the       *
*       LOCATE command, or by checking an index (PNMBR.NDX) of PRO:NMBR's      *
*       with the FIND command (which is the method used here).                 *
********************************************************************************

*       perform initial housekeeping and set options
erase
set talk off
set carry off
set colon off
set bell off

*       initialize memory variables
store ' ' to CHOICE
store ' ' to ANY:KEY
store '       ' to PRONUM

*       select database and set index
use PROPOSAL index PNMBR

*       main routine loop - continue until exit is requested
do while CHOICE () 'E'
     erase
     @  4,26 say 'Proposal File Maintenance'
     @  5,26 say '--------------------------'
     @  7,26 say '  A - add new proposal'
     @  8,26 say '  C - change proposal'
     @  9,26 say '  D - delete proposal'
     @ 10,26 say '  L - list proposals'
     @ 11,26 say '  E - exit'
     @ 13,26 say '  Choice ';
             get CHOICE picture '!'
     read

     do case
*       routine to add a new record to the database
        case CHOICE = 'A'
             erase
             @  6,14 say 'Please enter the proposal information as requested.'
             @  8,14 say 'Press (return) for Proposal Number when finished.'
             @ 23,50 say 'Press any key to continue ';
                     get ANY:KEY picture "X"
             read
             store T to MORE

*       continue as long as an proposal number is entered
             do while MORE
                erase
                store '       ' to PRONUM
                @  1,0  say 'PROPOSAL FILE -- ADD'
                @  2,0  say '--------------------'
                @ 10,10 say 'Proposal Number ';
                        get PRONUM picture "XXXXXXX"
                read
                if PRONUM()'       '
*       if a proposal number is entered, try to find if it already exists
                   find &PRONUM
                   if #=0
*       proposal number doesn't exist, so collect information and append record
                      append blank
                      replace PRO:NMBR with PRONUM
                      @ 12,10 say 'Proposal Name   ';
                              get PRO:NAME   picture "XXXXXXXXXXXXXXXXXXXXXXXXX"
                      @ 13,10 say 'Proposal Client ';
                              get PRO:CLIENT picture "XXXXXXXXXXXXXXXXXXXXXXXXX"
                      @ 14,10 say 'Proposal Fee    ';
                              get PRO:FEE    picture "########.##"
                      @ 15,10 say 'Other Dir Costs ';
                              get PRO:ODC    picture "######.##"
                      @ 16,10 say 'Proposal Status ';
                              get PRO:STATUS picture "X"
                      read

                   else
*       proposal number already exists, so give message and information
                      @ 10,40 say 'Duplicate Proposal Number!'
                      @ 11,42 say PRO:NMBR using "########"
                      @ 11,42 say PRO:NAME
                      @ 12,42 say PRO:CLIENT
                      @ 23,50 say 'Press any key to continue ';
                              get ANY:KEY picture "X"
                      read
                   endif
                else
                   store F to MORE
                endif
             enddo

        case CHOICE='C'
*         perform change proposal routine

        case CHOICE='D'
*         perform delete proposal routine

        case CHOICE='L'
*         perform list (display or print) proposals routine

        otherwise
*         either 'E' for exit or invalid repsonse

     endcase
enddo

*       perform final housekeeping before exit
use
release all
return
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
************     TIMECARD.PRG  --  Employee Timecard Batch Entry    **********
*        The TIMECARD.PRG routine performs the essential function of          *
*        batch entry of employee timecards into the TIMECARD database.       *
*        A second database, EMPLOYEE, is used in the process to verify        *
*        employee numbers.  The two files are related by employee number      *
*        as can be seen from a list of their structures.  Once entered,       *
*        timecards may be edited, deleted and listed prior to posting         *
*        batch to the master timecard file, TIMEMAST.  The TIMECARD batch     *
*        may be cleared after posting to the TIMEMAST file.                   *
*                                                                             *
*             TIMECARD.DBF                        EMPLOYEE.DBF               *
*        ----------------------              ----------------------         *
*        01   EMP:NMBR    N  04              01   EMP:NMBR  N  04            *
*        02   PRO:NMBR    C  07              02   EMP:NAME  C  25            *
*        03   PROP:HRS    N  04,01           03   HRLY:RATE N  05,02         *
*        04   PRES:HRS    N  04,01                                           *
*        05   NEGO:HRS    N  04,01                                           *
*        06   ENTRY:DATE  C 08                                               *
*                                                                             *
*        The routine begins by selecting the primary and secondary files.     *
*        Timecard information is entered with validation of Employee          *
*        Number from the EMPLOYEE file.  Checking the Employee Number         *
*        involves searching the EMPLOYEE file and can be accomplished in      *
*        several ways: manual search via the SKIP command, sequentially       *
*        reading through the file with the LOCATE command, or by FINDing      *
*        the Employee Number in the ENMBR index.  If the employee number      *
*        cannot be found in the EMPLOYEE file, an appropriate message is      *
*        displayed and the number may be reentered.                           *
*******************************************************************************
*      perform initial housekeeping and set options
erase
set talk off
set carry off
set colon off
set bell off

*      initialize memory variables
store ' ' to CHOICE
store ' ' to ANY:KEY
store '    ' to EMPNUM
store 0 to E:NUM

*      select primary and secondary databases and set index
use TIMECARD
select secondary
use EMPLOYEE index ENMBR
select primary

*      main routine loop - continue untill exit is requested
do while CHOICE <> 'E'
    erase
    @  4,26 say 'Timecard Batch Entry'
    @  5,26 say '--------------------'
    @  7,26 say 'A - add timecards'
    @  8,26 say 'C - change timecards'
    @  9,26 say 'D - delete timecards'
    @ 10,26 say 'L - list timecards'
    @ 11,26 say 'P - post timecards'
    @ 12,26 say 'E - exit'
    @ 14,26 say 'Choice ';
            get CHOICE picture '!'
    read


    do case
*      routine to add a new timecard record to the database
       case CHOICE = 'A'
            erase
            @  6,12 say 'Please enter the timecard information as requested.'
            @  8,12 say 'Press (return) for Employee Number when finished.'
            @ 23,50 say 'Press any key to continue ';
                    get ANY:KEY picture "X"
            read
            store T to MORE
*      continue as long as an employee number is entered
            do while MORE
               erase
               store 0 to E:NUM
               @  1,0  say 'TIMECARD FILE -- ADD'
               @  2,0  say '--------------------'
               @ 10,10 say 'Employee Number ';
                       get E:NUM picture "####"
               read
               if E:NUM<>0
*      if an employee number is entered, convert to a string value
*      and select the EMPLOYEE database and try to find it
                  store str(E:NUM,4) to EMPNUM
                  select secondary
                  find &EMPNUM
                  if #<>0
*      employee number is found, so display employee name for validation
                     @ 10,33 say EMP:NAME
*      select primary (TIMECARD) database so new record can be appended
                     select primary
                     append blank
                     replace EMP:NMBR with E:NUM
                     @ 12,10 say 'Proposal Number ';
                             get PRO:NMBR picture "XXXXXXX"
                     @ 14,10 say 'Proposal Hours  ';
                             get PROP:HRS  picture "##.#"
                     @ 16,10 say 'Presentn Hours  ';
                             get PRES:HRS  picture "##.#"
                     @ 18,10 say 'Negotatn Hours  ';
                             get NEGO:HRS  picture "##.#"
                     read
                  else
*      employee number is not found, so give message and continue
                     @ 10,40 say 'Invalid Employee Number!'
                     @ 23,50 say 'Press any key to continue ';
                             get ANY:KEY picture "X"
                     read
                  endif
               else
                  store F to MORE
               endif
            enddo
       case CHOICE='C'
*        perform change timecard routine
       case CHOICE='D'
*        perform delete timecard routine
       case CHOICE='L'
*        perform list (display or print) timecards routine
       case CHOICE='P'
*        perform posting (appending) of TIMECARD to TIMEMAST
*        this also could purge the batch TIMECARD file
      otherwise
*        either 'E' for exit or invalid repsonse
    endcase
enddo
*      perform final housekeeping before exit
use
release all
return
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
************     PREPORT.PRG  --  Proposal Costs Status Report     ************
*                                                                             *
*       The PREPORT.PRG routine provides the central Proposal Costs           *
*       Status report.  The report makes use of three database files          *
*       in deriving information to be presented in the report.  The           *
*       primary database is the employee TIMEMAST (timecard) file which       *
*       contains the cumulative work history of all employees.  This          *
*       database is accessed in order of Proposal Number (PRO:NMBR) and       *
*       timecard posting date (TC:DATE) via the index TCPDATE.NDX.  The       *
*       EMPLOYEE and PROPOSAL databases are alternately used as second-       *
*       ary databases and are accessed via the ENMBR.NDX and PNMBR.NDX        *
*       indexes, respectively.                                                *
*       TIMECARD.DBF            EMPLOYEE.DBF            PROPOSAL.DBF          *
*   ---------------------  ----------------------  --------------------      *
*   01  EMP:NMBR   N  04    01  EMP:NMBR   N  04    01  PRO:NMBR    C 07      *
*   02  PRO:NMBR   C  07    02  EMP:NAME   C  25    02  PRO:NAME    C 25      *
*   03  PROP:HRS   N  04,01  03  HRLY:RATE N  05,02  03  PRO:CLIENT  C 25      *
*   04  PRES:HRS   N  04,01                         04  PRO:FEE     N 10,02   *
*   05  NEGO:HRS   N  04,01                         05  PRO:ODC     N 08,02   *
*   06  ENTRY:DATE C 08                            06  PRO:STATUS  C 01      *
*                                                                             *
*       The routine begins by selecting the primary TIMEMAST database,        *
*       reading through the file in Proposal Number - Entry Date order        *
*       by means of the TCPDATE index.  For each new Proposal Number,         *
*       the secondary PROPOSAL database is accessed to retrieve needed        *
*       information for the report.  Likewise for each new Employee Num-      *
*       ber, the secondary EMPLOYEE database is accessed.  Access to          *
*       both secondary databases is made via the FIND command using the       *
*       PNMBR and ENMBR indexes.  A print line is constructed and sent        *
*       to the printer for each employee timecard entry in the TIMEMAST       *
*       database.                                                             *
*******************************************************************************
*     perform initial housekeeping and set options
erase
set talk off
set carry off
set colon off
set bell off
*  set printer on and perform other initial housekeeping
set print on
store '       ' to PRONUM
store '    ' to EMPNUM
store 0 to E:NUM
store '       ' to LASTPRO
store '    ' to LASTEMP

*     select database and set index
use TIMEMAST index TCPDATE
select primary

*     position at logical beginning of file -- based on index key value
*     database will be be read in sequential order
goto top
*     continue until an end of file is reached on the TIMEMAST file
do while .not. eof
    store PRO:NMBR to PRONUM
*     if proposal number is different from last one, select PROPOSAL database
*     as secondary and try to find proposal number
    if PRONUM()LASTPRO
       select secondary
       use PROPOSAL index PNMBR
       find &PRONUM
       if #()0
*      if proposal number is found, retrieve needed information
          store PRO:NAME to PRONAME
          store PRO:CLIENT to PROCLIENT
          store PRO:FEE  to PROFEE
          store PRO:ODC  to PROODC
          store PRO:STATUS to PROSTATUS
       else
*      if proposal number is not found, store message and clear values
          store 'Proposal Not Found' to PRONAME
          store ' ' to PROCLIENT
          store 0 to PROFEE
          store 0 to PROODC
          store ' ' to PROSTATUS
       endif
    endif

    store str(EMP:NMBR,4) to EMPNUM
*     if employee number is different from last one, select EMPLOYEE database
*     as secondary and try to find employee number
    if EMPNUM()LASTEMP
       select secondary
       use EMPLOYEE index ENMBR
       find &EMPNUM

       if #()0
*      if employee number is found, retrieve needed information
          store EMP:NAME to EMPNAME
          store HRLY:RATE to EMPRATE
       else
*      if proposal number is not found, store message and clear values
          store 'Employee Not Found' to EMPNAME
          store 0 to EMPRATE
       endif
    endif

* ---) routine to print detail line on report including check for 'new page'
* ---) would be implemented at this point
*      save current proposal and employee keys
    store PRONUM to LASTPRO
    store EMPNUM to LASTEMP
*      select TIMEMAST database and read next record
    select primary
    skip

    if PRO:NMBR()LASTPRO
* ---) routine to print subtotals/totals for proposal if next timecard
* ---) contains a new proposal number
    endif
enddo
* turn off printer and perform final housekeeping before exit
use
set print off
release all
return
```

□

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ULTRA UTILITIES

by Greg Estes

If you have enjoyed the Norton Utilities then you will appreciate the Ultra Utilities from FreeSoft. The FreeSoft Company will deliver to you an extensive set of diskette utility programs for free. The programs offer a wide variety of services for diskette files. If you teach any classes they would provide excellent tutorial material.

The Ultra Utilities will:

Display/Modify disk or file sectors
Copy disk or file sectors
Search for bytes in a disk or file sector
Modify sectors
Display/Modify disk parameters
Handle non-standard sectors
Format a standard track
Re-format a standard track with old data
Modify disk parameter table
Modify/Display radix
Complete directory utility (display, modify, erase)
Build a file from scratch
Recover a file, usually automatically
Produce a diskette map of files by sector

One of the features I like best is a complete printout of files broken down by track and sector. When you want to reconstruct a file. Just printout a complete copy of the disk's sector usage, hold it in your hand, and piece together a file with your road map sitting right there on the table. The printout looks like this:

```
Map of diskette space allocation

Meaning of codes shown below:

01---)IBMBIO  .COM      02---)IBMDOS  .COM      03---)COMMAND .COM
04---)BRACE   .BAS      05---)AUTOEXEC.BAT      06---)SC      .COM
07---)RAM     .COM      08---)SCROLL  .COM      09---)FK      .COM
10---)FK      .KEY      11---)FORMAT  .COM      12---)VC      .COM
13---)MONSHRIN.BAS      14---)GR      .KEY      15---)VEDIT   .INI
16---)DK      .COM      17---)BUGS!   .SCR      18---)C       .COM
19---)DDEL    .COM      20---)CINPUT  .C        21---)SDIR    .COM
22---)AUTOEXEC.BAK      23---)KILLSPAC.EXE      24---)WSFIX   .EXE
25---)BASICA  .COM      26---)CURT    .BAK      27---)AUTHOR  .
28---)CURT    .         29---)PJDISCNT.         30---)CORRD   .BAK
31---)END     .BAS      32---)CORRO   .BAT      33---)FORUP   .PRG
34---)CHKPRT  .PRG      35---)ONESPAC .BAS      36---)WSFIX   .BAS
37---)GMC     .BAS      38---)OKIDEMO .BAS      39---)CASSADDR.
40---)KILLSPAC.BAS      41---)PJ1     .         42---)JET     .COM
43---)CORRO   .         44---)CONFIG  .SYS      45---)AUTOEXEC.BAA
46---)BUGS    .EXE      47---)COLOR   .COM      48---)MONO    .COM
51---)FILECONV.BAS      52---)MONCLK  .COM      53---)FKBAT   .BAT
54---)MONBLK  .COM      55---)AST     .BAT      56---)DOS     .BAK
57---)DOS     .BAT
LCK---) Locked out, ORP---) Orphan,  EMP---) Empty, SYS---) System.
```

```
                        Double Sided Diskette - Side 0
                        ------------------------------
Track   Sect-1 Sect-2 Sect-3 Sect-4 Sect-5 Sect-6 Sect-7 Sect-8
-------------------------------------------------------------------
00---)  (SYS)  (SYS)  (SYS)  (SYS)  (SYS)  (SYS)  (SYS)  (SYS)
01---)  =002=  =002=  =002=  =002=  =002=  =002=  =002=  =002=
02---)  =003=  =003=  =003=  =003=  =003=  =003=  =045=  =045=
03---)  =007=  =007=  =007=  =007=  =008=  =008=  =009=  =009=
04---)  =011=  =011=  =011=  =011=  =011=  =011=  =013=  =013=
05---)  =012=  =012=  =012=  =012=  =012=  =012=  =012=  =012=
06---)  =012=  =012=  =012=  =012=  =012=  =012=  =012=  =012=
07---)  =016=  =016=  =022=  =022=  =031=  =031=  =026=  =026=
08---)  =020=  =020=  =020=  =020=  =021=  =021=  =021=  =021=
09---)  =044=  =044=  =004=  =004=  (EMP)  (EMP)  (EMP)  (EMP)
10---)  (EMP)  (EMP)  (EMP)  (EMP)  (EMP)  (EMP)  (EMP)  (EMP)
11---)  (EMP)  (EMP)  (EMP)  (EMP)  (EMP)  (EMP)  (EMP)  (EMP)
12---)  =023=  =023=  =023=  =023=  =023=  =023=  =023=  =023=
13---)  =023=  =023=  =023=  =023=  =023=  =023=  =023=  =023=
14---)  =023=  =023=  =023=  =023=  =023=  =023=  =024=  =024=
15---)  =024=  =024=  =024=  =024=  =024=  =024=  =024=  =024=
16---)  =024=  =024=  =024=  =024=  =024=  =024=  =024=  =024=
17---)  =024=  =024=  =024=  =024=  =024=  =024=  =024=  =024=
18---)  =025=  =025=  =025=  =025=  =025=  =025=  =025=  =025=
19---)  =025=  =025=  =025=  =025=  =025=  =025=  =025=  =025=
```

The documentation contains quite a bit of technical information about diskettes and files, standard formats as well as non-standard formats. A blow-by-blow guide through the radix, disk parameter tables, and formatting options is included.

For a copy send a formatted DS diskette with postage paid return mailer to:

FreeSoft Company (Ultra Utilities)
P. O. Box 27608
St. Louis, MO 63146

□

INTERNATIONAL PC OWNERS
IPCO
"The international user group for the PC"
- Software library of 175+ programs @ $3 each
- Datacount® General Ledger/Manual $35
- Volkswriter® $125
- Bimonthly newsletter 40+ pages
- Worldwide information exchange
- "Freeware"

FOR FREE CATALOG WRITE:
INTERNATIONAL PC OWNERS, INC.
ATTN: FREE CATALOG
P.O. BOX 10426
PITTSBURGH, PA 15234

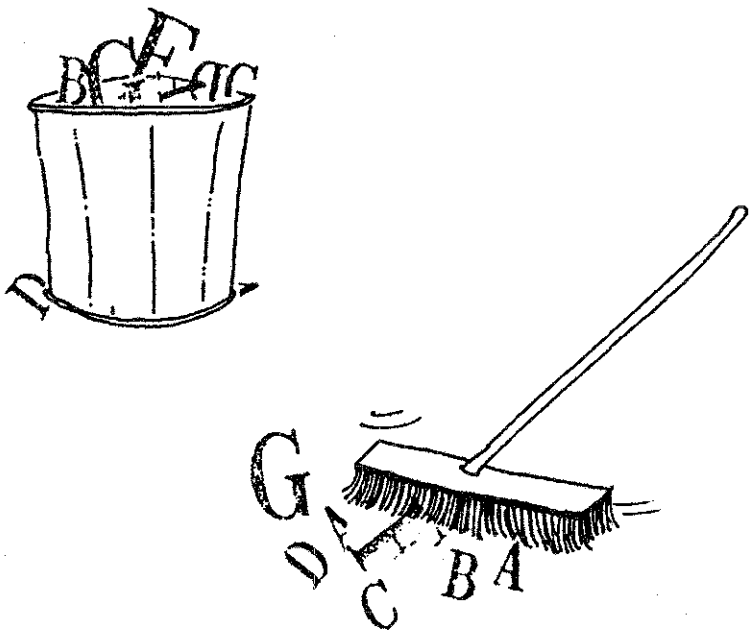
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SOFTWARE REVIEWS

PUNCTUATION AND STYLE

a review by Richard Wanderman

Punctuation and Style, $125.
The Word Plus, $150.
Oasis Systems
2765 Reynard Way
San Diego, CA 92103
(619-222-1153)



Writing with a computer and word processing software revolutionizes the writing process: fast and painless editorial changes, no more re-writing whole drafts, printing separate from writing, plus proofreading programs that check spelling, punctuation and style—a writer's dream!

Punctuation and Style (P & S) is a proofreading program made up of two parts: CLEANUP and PHRASE. They are used independently for processing your documents.

CLEANUP will find punctuation errors, capitalization errors, odd numbers of quotes, brackets and toggles, numeric errors, abbreviation errors and repeated words.

PHRASE will find 500 commonly misused words or phrases and make suggestions for revising them. It also finds passive voice sentences and helps turn them into active voice sentences by highlighting prepositions and forms of the verb "to be"

A potential problem with programs that use conventional patterns and ignore context (how else could a proofreader work?) is that they reduce the individuality in your writing and homogenize it. No one's going to shoot us if we put four spaces after a period, write everything in passive voice, or write in lists instead of sentences. I use P & S selectively, as a reference tool. I want my writing to meet some minimum standards of form so it will be read and taken seriously by others. But I don't want to be locked into a rigid way of expression that doesn't allow for rule breaking and changing the form to fit the purpose/context. There are many different ways of putting a sentence together, thank God! P & S, however, is far from rigid. You can customize it to fit your writing style and personal conventions. This customization option makes P & S a powerful tool for improving your writing.

It is menu driven. Minimal reading need be done before using the programs. The documentation is excellent. Put the program in one drive. Put the file to be checked in the other. Call up either program by typing "CLEANUP" or "PHRASE," and the filespec to be checked. That's it, you're off. Both programs will mark the revision suggestions in your file. CLEANUP and PHRASE do not have an automatic replace feature. You can't type in the needed corrections while using P & S and have them inserted automatically. They both make backups of the file being checked. Make sure you've got the disk space to handle this automatic doubling of file size.

Customization of CLEANUP and PHRASE is done by including certain "switch" commands when calling up the programs or creating new pattern files. The customization process is explained clearly in the documentation. You might want to have multiple copies of the program for checking different types of writing, one for formal writing with all the strict conventions switched on, and one for informal writing with a looser set of conventions.

CLEANUP:

Some of CLEANUP's default patterns are based on conventions that aren't popular. For instance, the program flags the use of one space (it considers two correct) after the end of a sentence. It flags the use of an ellipsis "..." with no spaces between the dots, (it

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

demands a space: ". . ."). This isn't a cut-and-dried situation, it's a matter of convention. I've just been thumbing through some books and magazines: THE ELEMENTS OF STYLE, The New York Times Book Review, The Atlantic and The New Yorker. None of them use two spaces after a period, or spaces in ellipses. That doesn't make one convention right or wrong, it just tells me that I shouldn't necessarily convert to the ones used in the program, thinking that they're universal.

Note: The screen messages in Oasis Systems' other product, THE WORD PLUS, have no spaces in the ellipses. Hmmm.

Conventions aren't a problem if you're secure with your writing style. The problem with a convention like "two spaces after a period" arises when you have to deal with CLEANUP flagging every one of them. This makes the customizing option essential. I created an "option file" which tells the program to stop flagging "one space after a period" and "unspaced ellipses." I can either have that option file present on the disk, or leave it off depending on the conventions I want used for my punctuation. This is very powerful stuff. I appreciate the chance to get at the innards of the program.

Most people make few clerical mistakes with basic sentence punctuation like periods and commas. But there is a real possibility for clerical error when using quotation marks, parentheses, and control-character toggles in programs like WordStar. CLEANUP will flag unbalanced punctuation and control characters. I wonder how it would work on a LISP program?

CLEANUP will also flag abbreviations that contain embedded periods that aren't listed in its abbreviations file. You can update this file (with "D.D.S." perhaps) to include those that you use often and don't want flagged. You can also update a file that contains exceptions to capitalization rules.

PHRASE:

Proofreading programs do not consider context (yet) and this does present certain difficulties. When using PHRASE, I ignore its suggestions at least 75% of the time: they're out of context. But when the suggestions fit, they're very useful.

You may use "switches" to invoke PHRASE and tell it how and where to store the suggestions. It can place them at the end of the offending sentence, at the end of the paragraph or you can send them to the printer. This eliminates the problem of trying to remember what the flagged offense was, but it means that you have to delete PHRASE'S suggestions from your file as you deal with them. You can tell PHRASE to just put brackets around problem words and sentences. This is its default mode. If you want in-context suggestions you must ask for them. PHRASE has a switch which tells it to go through the whole file, automatically marking all problems without interruption.

You can also create and use your own "pattern files": dictionaries of words and phrases that you might be over-using. This ability to customize makes PHRASE more personally appropriate. You may have multiple versions of the program: one that eliminates certain overworked phrases, and another that has a coarser filter which lets some cliches through.

Of the two parts of P & S, CLEANUP will probably see more use than PHRASE. That the two parts of the program operate independently of one another means that CLEANUP can become a regular part of your writing process, like checking spelling.

The ability to build and manipulate different personalized versions of P & S is wonderful. Its underlying assumptions about how people learn are shared by Oasis Systems' companion spelling checker, The Word Plus. Both programs are very flexible tools, easy to use, inexpensive and they work with a variety of word processors.

C SCREEN EDITOR

CSE: A full-screen text editor written in C

$60 Includes the complete source code

- Powerful command set includes cursor control, find/replace, block move, file inclusion, and nested macro commands.
- Full-screen editing, including cursor control, insert/overtype, single and multiple line insertion and deletion, and forward and reverse scrolling.
- Commands to perform large-scale operations on the text buffer, such as find, find and replace, move the text pointer, read input, and write output.
- A macro facility which allows grouping of a set of commands for repeated execution.
- Ability to work on files larger than the text buffer size.
- Complete configurability of control sequences, such as cursor keys and editor control strings, to allow customization of CSE for a specific terminal. CSE is configured to use special editing and function keys on the IBM PC.

NMD Northwest Microsystem Design

P.O. Box 10853 • Eugene, OR 97401 • (503) 484-7129

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ LIBRARY

LAWYERS' COMPUTER BOOK

"Computer Power for your Law Office" is a new offering from Sybex Press. The author, Daniel Remer, has also written an excellent definitive book on software law. If this book is even close to the quality of his "Legal Care for your Software" release, it should be a valuable addition to your legal practice. Contact:

Sybex Press
2344 6th St.
Berkeley, CA 94710
(415-848-8233)

PASCAL/ADA/MODULA-2

If Pascal, Ada or Modula-2 are of interest, you should enjoy "The Journal of Pascal and Ada" from Utah. The Journal gets better and better with every issue. $14 a year for subscribers:

Pascal and Ada Journal
PO Box 384
Orem, UT 84057

COMPUTER MAGAZINE XREF

A bi-weekly publication lists the table of contents pages from a number of computer, electronic and telecommunication magazines. If you wish to keep up with every article on your favorite topic then it would be an improvement over subscribing to several hundred magazines. Write for a sample issue.

Computer Contents
PO Box 3014
Northbrook, IL 60062
(800-323-5354)

PUBLIC DOMAIN BOOKLET

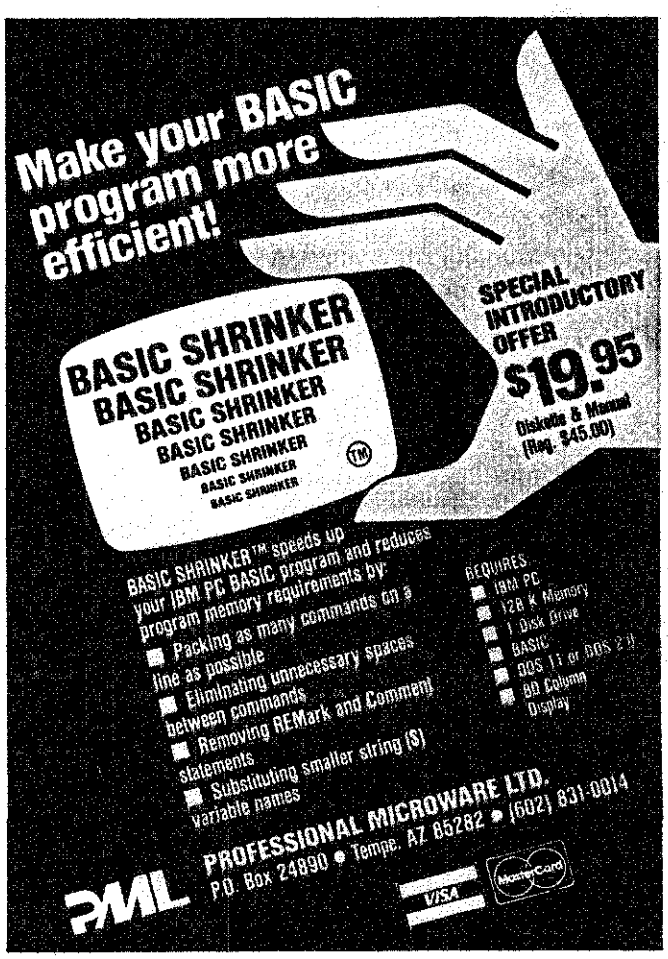
Have you been wondering where you can find a comprehensive list of the available IBM PC public domain software? You can find it by requesting a copy of the "Public Domain" booklet from PC Software Interest Group. The booklet costs $2.95 plus $1 for handling. You may also order any of the software from them. A set of 10 diskettes is offered for $59 or your group may purchase their entire collection of 75 diskettes for $439.

PC Software Interest Group
1556 Halford Ave. Suite #130R
Santa Clara, CA 95051
(408-247-6303)

INTEL YELLOW PAGES

The Intel Corp publishes an extensive listing of the products running on their computers. 2,000 products and technical services are mentioned. If you are a programmer, you may receive a copy free. From XENIX to MS-DOS, RMX-86 to CP/M-86 and everything in between. Write and ask for:

Intel Yellow Pages
Mail Stop SC6-295
3065 Bowers Ave.
Santa Clara, CA 95051
(408-987-5057)



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MISC.

BASIC TO C

A new UNIX software house, Software Manufacturers Inc., has developed a package which will take an application written in BASIC and translate it into C to allow it to run under UNIX. Their S-Tran converter allows companies to benefit from a move to UNIX without sacrificing their BASIC application systems. It will currently translate Microsoft's MBASIC or Oasis BASIC to C with other versions of BASIC soon on the way. Modifications can be made in the original BASIC and retranslated, or maintenance can be performed in the C version. Additional features supported are ISAM and terminal independence. CHAIN with COMMON, RSET, LSET, MID$, RPAD, LPAD, LINE INPUT, PRINT USING, ON ERROR GOTO, and other functions are provided. The company markets accounting packages for UNIX-based systems.

S-Tran ($1500)
Software Manufacturers Inc
Torrance, CA
(213-316-2220)

PROGRAM BUILDER - UNIX STYLE

Polytron offers a new tool for programmers, POLYMAKE. It is an enhanced version of the UNIX Make program that will run under MS-DOS. When a change is made to any module in a software system, POLYMAKE can automatically reconstruct those files dependent on the altered file. POLYMAKE will invoke your compiler, assembler, linker, librarian, or do whatever is necessary to bring all dependent files up to date. Date and time stamps are compared accordingly. It runs on an IBM PC. Available March 1, 1984.

POLYMAKE ($99)
Polytron Corp
PO Box 787, DS 2-306
Hillsboro, OR 97123
(800-547-4000)

AST UPDATE

If you use an add-on board from AST, you know they sell superior hardware. IBM apparently thinks so too. They will be selling AST boards (MegaPlus II and SixPacPlus) at their product centers. IBM national accounts dealers and value added dealers will also carry them. Definitely the stamp of approval for a first-class company that develops products only for the PC.

RESET BUTTON FANS

Answer Software Corp has announced an add-in board for the IBM-PC. It provides a state of the art debugging and performance tool for software developers. The Debugger offers symbolic debugging, both high level and OBJ formats, full screen display containing program history, current status with future instructions, hardware breakpoints, memory usage, module duration, inter-module calling, linkage tracking, external hardware synchronization, and trigger output to oscilloscopes or logic analyzer. PRESS A BUTTON to recover from a program crash. A demo diskette is available for $20, refundable with a full purchase. For additional information contact:

Edward Tang
PDT-PC Tool
Answer Software Corp
20863 Stevens Creek Blvd, B2-C
Cupertino, CA 95014
(408-253-7515)

DOCTOR TAKE NOTE

The State of New York's fiscal agents for Medicaid are now accepting data on IBM PC diskettes for processing of plans. It is reported that Blue Cross expects to be accepting PC disks in about three months. For more information call Joyce Burich at 212-577-9684. (This note from IPCO).

8087 APPLICATION NOTES

Field Computer Products is offering a number of reprints from Intel concerning the 8087, besides additional 8087 products (assembly 8087 macros and a Pascal primitive library). The 8087 booklets are:

The 8087 Numeric Data Processor, 114 pages, $12
Getting Started with the Numeric Data Processor, 60 pages, $6
Using the 8087 in a Small Business Computer, 22 pages, $4
Three Dimensional Graphics Applications of the 8087, 50 pages, $5

The ASM and Pascal products are $50 each. Several other Epson printing enhancers are available too.

Field Computer Products
909 North San Antonio Road
Los Altos, CA 94022
(415-949-3457)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

COBOL SCREEN BUILDER

Qwik Screen Builder is a COBOL source code generator for interactive displays designed for the IBM COBOL Compiler. Paint the screen you want using a full screen editor or copy in existing screens for modification. You define the field attributes. Both the SCREEN SECTION and the WORKING STORAGE SECTION are prepared. Several samples are provided plus a 100 page manual. The price is an unbelievable $50 with a money-back guarantee.

Qwik Screen Builder ($50)
The Computer Works
PO Box 1371
Coeur d'Alene, ID 83814
(208-667-6152)

DISKETTE UTILITY FREEWARE

Three IBM PC disk utilities are available under the Freeware program. They perform many of the functions of the Norton Utilities as well as a number of others. They examine sector information, reclaim erased files (normally automatically), allow a full screen delete by request option. The Ultra-Utilities, as they are called, offer an excellent tutorial aid for learning your way around diskettes. You may request a printout of the track and sector usage on a diskette by file name and number. Request a copy today! Send a formatted double sided diskette or two single sided ones, plus a postage paid return mailer to:

FreeSoft Ultra-Utilities
PO Box 27608
St. Louis, MO 63146

FORTH FOR FREE

The San Francisco IBM PC Users group has a Freeware version of the FORTH language. FORTH is available on their club disks #40 and #41. The documentation can be purchased from Mountain View Press. For the diskettes, write:

SF IBM PC Group
4411 Geary Blvd. #33
San Francisco, CA 94118
Librarian: Charlie Vella, 415-387-2315

Forth Documentation
Mountain View Press
PO Box 4656
Mt. View, CA 94040

SOFTWARE DEVELOPERS!

Polytron Corporation, an Oregon based company specializing in software for technical professionals, has announced its intention to market software developed by independent software writers. Polytron is currently marketing PolyLibrarian, an Object Module Librarian for MS-DOS.

According to Richard Kinnaird, President, "We decided to pursue expanding our product line with high-quality software developed by outstanding software engineers outside our own company. We are looking for excellent MS-DOS products that assist programmers in the development or maintenance of software written in C, Pascal, Fortran or Assembler."

When asked if he preferred submissions to be written in any particular language, Richard stated "We use C as our in-house development language, but Pascal, Assembler and Fortran are also perfectly acceptable."

According to Doug Root, Director of Marketing, "We are not interested in products for business or home use. We specialize in high-quality productivity-enhancement software for engineers, programmers and scientists. We will provide marketing, documentation, manufacturing and development help to make sure outside products meet our standards. We are looking for people with products already developed as well as those with new ideas."

Software authors interested in exploring opportunities with Polytron can call 503-648-8595 or write

Richard Kinnaird,
Polytron Corp.
PO Box 787
Hillsboro, OR 97123

MULTI-PHONE LINE CONNECTOR

Data Acquisition Inc. has a number of accessory products for personal computers: keyboard extension cables, joy-stick extension cables, etc. One product they sell is an adapter for multiple phone line plugs. The adapter fits over the large, oblong outlet and turns it into a normal single line phone jack with a knob that switches among the five lines. It allows you to easily hook up your modem to any of the five lines. ($34.95)

Data Acquisition Inc.
20326 NE 16th Place
North Miami Beach, FL 33179
(305-652-2623)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MORE PUBLIC DOMAIN

During the past 18 months I have collected over 1600 public domain programs. These cover the following general topics:

- Educational Items
- Games
- Graphics
- Mathematics, Statistics
- Text Editing
- Data Communications
- BASIC Programming Aids
- Many Utilities for controlling printing, keyboards, CRT's, etc.

I am very interested in trading with other readers for items which they have written themselves. They can contact me at Sunmark's address, or on the following networks

- The Source - STG422
- Compuserve - 70426,33
- Delphi - Vince
- Plato - pca/vince e heiker/lime 1

I would appreciate your putting some type of notice under the MISC. banner for the next issue or two. This will help me to continue to expand my library and will also help your many readers obtain free software and examples of various kinds of programs.

Thank you.

Vince E. Heiker, MIS Director
The Sunmark Companies, Corporate Offices
10795 Watson Road
St. Louis, MO 63127
(314-822-2800)

VOLITION MODULA-2

A very comprehensive version of Modula-2 is now available from Volitions Systems. The language is part of a software system based on version II of the UCSD Pascal Operating System. It can run on a 64K machine. The system offers low-level hardware access, real-time control, concurrent processing, and type-secure separate compilation. 8087 support and interrupt handling are also included.

Modula-2 ($595)
Volition Systems
PO Box 1236
Del Mar, CA 92014
(619-481-2286)

XERGO/PMATE

XERGO is a full-featured word-processor built around a PMATE base. XERGO includes a complete macro facility very similar to PMATE but slightly different. You need not learn to write macros, XERGO already supplies the text editing and printing features you will most likely need. Should you later find a particular feature you desire, or want to alter one that currently functions, you are free to do so using the macro language. Ten buffers are available for cutting and pasting. The full price ($335) includes XERGO with numerous tutorial files, large manual, and printer configuration possibilities.

If you already have PMATE and desire the printing and additional features of XERGO, then you can order an unbundled version designed to interface with it. The price is $170 for the interface part. For more detailed information contact:

XERGO
Landmark Software
1142 Pomegranate Court
Sunnyvale, CA 94087
(408-733-4032)

MAILING LISTS

Going to do some mass mailings? Send for a copy of Irv Brechner's Targeted Mailing catalog to get an idea of the possibilities. He is putting together a Vertical Market report to outline the potential of this untapped area for software developers.

Irv Brechner
Targeted Marketing Inc.
Box 453
Livingston, NJ
(201-731-4382)

C-SYSTEMS

The Run Time Support Library is now included in the current release of C-Systems C compiler. They are the only ones that have an interpretive style debugger. DOS 2.0 is supported as are extended memory addressing, typedefs, and castes. The compiler is $195 and the Language Debugger is also $195.

C-Systems
PO Box 3253
Fullerton, CA 92634
(714-637-5362)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PROGRAMMER'S JOURNAL SUBSCRIPTION FORM

1. Current occupation or job title: _____
   Income: ☐ under $25,000 ☐ $25,000 to $50,000 ☐ over $50,000
2. Education: ☐ BA ☐ MA ☐ Ph.D.
3. What brand of microcomputer(s) do you own? _____
   _____
4. What brand of microcomputer(s) do you use at work? _____
   _____
5. Which programming languages do you use regularly? _____
   _____
6. What purchases are you planning in the next six months? ☐ hardware ☐ software ☐ peripherals
   Describe: _____
7. Which articles or features in *Programmer's Journal* do you find the most useful?
   _____
8. Which articles or features in *Programmer's Journal* do you find the least useful?
   _____
9. What additional departments or topics would you like to see added to the *Programmer's Journal?* _____
   _____

Include your:

Name _____ Address _____

Company _____ City _____

Telephone _____ State _____ Zip _____

☐ 1 year bi-monthly subscription rate:
$24 U.S. ($30 Canada, $40 other)

Total (including back issues) $_____

☐ Check enclosed ☐ Bill me

☐ Charge my Bankcard Expiration date _____

No. _____

☐ Send me the following back issues at $4.00 each:

Vol. 1, No. 1 ____ Vol. 1, No. 4 ____
Vol. 1, No. 2 ____ Vol. 1, No. 5 ____
Vol. 1, No. 3 ____ Vol. 1, No. 6 ____

Signature _____

Send to:

PROGRAMMER'S JOURNAL
P.O. Box 30160
Eugene, OR 97403
Ph. (503) 345-3043

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SWIG©

SOFTWARE WRITERS INTERNATIONAL GUILD

SCHEDULED SWIG ACTIVITIES & MEMBERSHIP BENEFITS

(1) $10,000 PROGRAMMING CONTEST (Members only)

(2) NATIONAL COMPUTER WEEK (May 11 - May 20, 1984)

(3) ANNUAL CONFERENCE AND SOFTWARE AWARDS CEREMONY (During National Computer Week)

(4) CONSULTANT REGISTRY (With computer store referral system for customized software)

(5) JOB PLACEMENT SERVICE (Free to individual members, fixed maximum fee to companies)

(6) FREE SEMINARS & MEETINGS LOCALLY

(7) SOFTWARE LIBRARY LENDING & EXCHANGE SERVICE (Professional quality assemblers, utilities, games, etc.)

(8) SOFTWARE LOCATION SERVICE (For companies & individuals-if it exists, **SWIG** will find it. If not, see #9)

(9) SOFTWARE DEVELOPMENT SERVICE (From novice to scientist, **SWIG** members can work on any project-from applications to games to R&D)

(10) LEGAL SERVICE

(11) AGENT (**SWIG** can represent you in sales to software publishers)

(12) 24 HOUR - 7 DAY BULLETIN BOARD SYSTEM (BBS) ACCESSIBLE BY COMPUTER FREE

(13) AND MORE!!!!

THE LARGEST PAID MEMBERSHIP PROGRAMMERS GUILD - OVER 5,000 MEMBERS WORLDWIDE!!

MEMBERSHIP APPLICATION FOR SOFTWARE WRITERS INTERNATIONAL GUILD

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

PHONE # ( ) _____

- CLASSIFICATION:

  ☐ NOVICE ☐ BEGINNER TO ADVANCED

  ☐ ADVANCED WITH ON THE JOB EXPERIENCE ☐ RESEARCH/SCIENTIST

- WHAT EQUIPMENT DO YOU HAVE EXPERIENCE WITH &/OR ACCESS TO &/OR PLAN TO BUY?

  ☐ MAINFRAME ☐ MINI ☐ MICRO ☐ DESIGN/R&D

  BRAND NAME(S): ☐ IBM ☐ XEROX ☐ APPLE ☐ TI

  ☐ COMMODORE ☐ RADIO SHACK ☐ ATARI ☐ OSBORNE

  ☐ TIMEX/SINCLAIR ☐ NORTH STAR ☐ HEWLETT PACKARD

  ☐ OTHER _____

- AREAS OF INTEREST:

  ☐ DATA PROCESSING ☐ BUSINESS APPLICATIONS ☐ GRAPHICS

  ☐ LEGAL ☐ VOICE ☐ MEDICAL ☐ APPLIANCE (HOME) CONTROL

  ☐ ROBOTICS ☐ GAMES ☐ MUSIC ☐ R&D ☐ OTHER _____

- MEMBERSHIP ACTIVITIES AND SERVICES OF INTEREST:

  READ THE LIST ON THE LEFT AND CIRCLE THE NUMBERS BELOW THAT APPLY.

  1 2 3 4 5 6 7 8 9 10 11 12

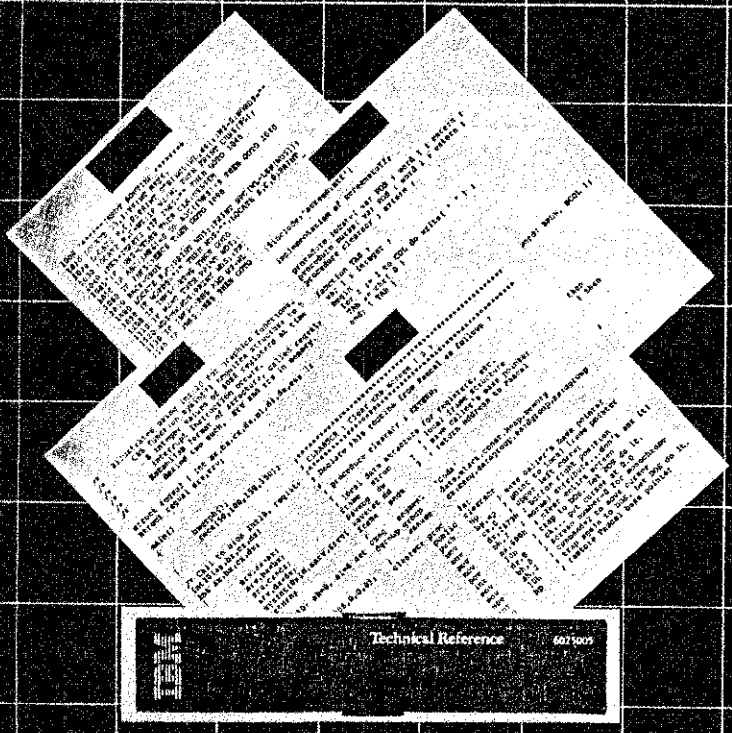
☐ I HAVE ENCLOSED $20 ANNUAL MEMBERSHIP FEE ☐ CK ☐ MO
(MAKE CHECK PAYABLE TO: **SWIG**)

RETURN TO: **SWIG**
P.O. BOX 87
STONY POINT, NEW YORK 10980
(914) 354-5585

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

programmer's journal

The Resource Journal for IBM PC Programmers



Volume 1 Number 2 March/April 1983

programmer's journal

The Resource Journal for IBM PC Programm

Volume 1 Number 6

programmer's journal

The Resource Journal for IBM PC Programmers

Volume 1 Number 2

programmer's journa

The Resource Journal for IBM PC Programme

"I own the software."

"No, it belongs to me!"

"Look, I wrote it."

"But I paid for the development."

"You can't be serious."

"I've already sold the marketing rights."

Volume 1 Numbe

PROGRAMMER.ORG
ELECTRONIC REPRODUCTION PROHIBITED